VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Cyklonavigační systém

Bike Navigation System

2013 Bc. Jakub Sviderek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Sviderek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Cyklonavigační systém
Bike Navigation System

Zásady pro vypracování:

Cílem práce je vytvořit cyklo-navigační systém, který se bude skládat z klienta určeného pro platformu Android pro mobilní telefony a serverové části. Serverová část umožní vytvářet a editovat, zveřejňovat trasy jednotlivých uživatelů systému. Mobilní klient pak umožní stahování tras, monitorování průjezdu trasy a navigaci trasou.

Funkce mobilní aplikace:
1. Zobrazení grafu rychlostí a převýšení trasy.
2. Zobrazení přehledových informací (doba jízdy, ujeté kilometry, zbývající kilometry, atd.).
3. Navigace po trase a sledování odchýlení od trasy.
4. Zobrazení trasy na mapě s vyznačením sklonu trasy.
5. Stažení trasy ze serveru.
6. Záznam trasy a její nahrání na server.

Funkce serverové části:
1. Uložení, vytvoření, editace, zobrazení, stažení tras.
2. Správa uživatelů systému.
3. Webové rozhraní se zobrazování tras v mapě.
4. Ukládání statistických informací o průjezdu tras.
5. Kategorizaci tratí dle kritérií (pozice (v okolí zadaného bodu mapy, okres, délka, obtížnost, typ kola, pro které je trasa vhodná, hodnocení uživatelů)).

Práce musí obsahovat:
1. Návrh aplikace v jazyce UML (Unified Modeling Language).
2. Implementaci aplikace.
3. Dokumentaci aplikace (programátorskou, uživatelskou).

Seznam doporučené odborné literatury:

[1] STEELE James; TO Nelson, The Android Developer's Cookbook: Building Applications with the Android SDK. 1. edition. UK: Addison-Wesley Professional, 2010. 400 s. ISBN 978-0321741233
[2] ESPOSITO Dino, Programming Microsoft ASP.NET 4. 1. edition. UK:
Microsoft Press, 2011. 992 s. ISBN 978-0735643383

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2013 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Rád bych na tomto místě poděkoval svému vedoucímu Ing. Davidu Ježkovi, Ph.D. za cenné rady, připomínky, odbornou pomoc a trpělivost při vytváření této diplomové práce.

Abstrakt

Úkolem diplomové práce je implementace cyklonavigačního systému. Cyklonavigační systém se skládá ze dvou částí. První část je mobilní aplikace určená pro platformu Android sloužící pro záznam tras a jako navigace. Druhá část je informační systém sloužící pro sdílení tras. Textová část práce je rozdělena do několika částí. První dvě části jsou věnovány popisu mobilní aplikace, první část popisu funkcí a druhá část implementaci. Ve třetí části jsou popsány funkce informačního systému. Čtvrtá část je věnována implementaci informačního systému. V poslední části je popsána komunikace mezi mobilní aplikací a informačním systémem.

Klíčová slova: diplomová práce, navigace, cyklistika, cyklonavigační systém, cyklonavigace, Android, ASP.Net, informační systém, WCF služby

Abstract

The task of master thesis is implementation of a cycling navigation system. Cycling navigation system is divided into two parts. The first part is a mobile application for the Android platform. Aplication serves to record routes and as a navigation. The second part is an information system. The information system is used for sharing routes. The text part is divided into several parts. The first two sections are devoted to the description of mobile application, the first part of the description of functions and the second part of the implementation. The third section describes functions of the information system. The fourth section is devoted to implementation of the information system. The last section describes communication between mobile application and information system.

Keywords: master thesis, navigation, cycling, cycling navigation system, cycling navigation, Android, APS.Net, information system, WCF services

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Úkolem této diplomové práce bylo naimplementovat cyklonavigační systém. Práce se skládá ze dvou částí. Z mobilní aplikace a z informačního systému.

První část je mobilní aplikace určená pro platformu Android. Součástí aplikace je databáze tras, k jednotlivým trasám lze ukládat informace o jízdě. Naměřené informace lze přehledně zobrazit do několika grafů. Jednotlivé trasy je možné zobrazit na mapě s vyznačenou nadmořskou výškou, aktuální rychlostí nebo sklonem trasy, na mapě lze také zobrazit navigační body trasy. Mobilní aplikace slouží také jako navigace, uživatel je po trase naváděn pomocí zvukových hlášení, a také se na obrazovce zobrazuje směr aktuálního a následujícího navigačního bodu, navíc je aplikace v režimu jízdy schopna přehrávat skladby, které jsou přidány v playlistu. Tato část částečně vychází z mé bakalářské práce. Aplikace je oproti bakalářské práci zcela předělána. Všechny důležité operace jsou prováděny ve službě, tím pádem nemůže dojít k jejich přerušení z důsledku nedostatku paměti. Také je upravena databáze a způsob vykreslovaní grafů, nyní jsou grafy vykreslovány v závislosti na velikosti obrazovky mobilního zařízení, na kterém je aplikace spuštěna. V aplikaci jsou přidány dvě nové sekce. Sekce Playlist, která slouží pro přidání skladeb přehrávaných v režimu jízdy, a sekce Synchronizace, sloužící pro komunikaci s informačním systémem.

Druhá část je informační systém sloužící pro sdílení jednotlivých tras mezi uživateli, a také pro vytváření nových tras a jejich úpravu. Informační systém je implementován v ASP.Netu, pro komunikaci mezi informačním systémem a mobilní aplikací jsou použity WCF služby. Pro posílání tras mezi mobilní aplikací a informačním systémem musí být uživatel v informačním systému zaregistrován. Uživatel může v informačním systému vytvářet nové trasy. U stávajících tras lze upravovat jejich souřadnice i navigační body. Uživatel také může trasy, které má stažené z mobilní aplikace nebo si je vytvořil pomocí informačního systému, zveřejnit. Po zveřejnění trasy si tuto trasu mohou jiní uživatelé přidat ke svým trasám a dále ji upravovat.

Textová část diplomové práce je rozdělena na pět částí. V prvních dvou částech je popsána mobilní aplikace. V první části jsou popsány všechny funkce mobilní aplikace, druhá část je věnována popisu implementace mobilní aplikace, tato část je doplněna výpisy zdrojového kódu. Ve třetí a čtvrté části je popsán informační systém. Ve třetí části jsou popsány všechny funkce informačního systému, čtvrtá část je věnována popisu implementace informačního systému, tato část je také doplněna výpisy zdrojového kódu. V poslední části je popsána komunikace mezi mobilní aplikací a informačním systémem, tato část je doplněna sekvenčními diagramy.

2 Popis mobilní aplikace

Diplomová práce se skládá ze dvou částí: z mobilní aplikace a z informačního systému. V této části budou popsány všechny funkce mobilní aplikace z uživatelského pohledu.

2.1 Zadání

Hlavním úkolem mobilní aplikace je navigace po trase, sledování odchýlení od trasy a zobrazení přehledových informací (aktuální rychlost, čas jízdy a ujetou vzdálenost), dále zaznamenává statické informace o jízdě, jako jsou souřadnice projížděné trasy, nadmořská výška a aktuální rychlost. Dále z naměřených hodnot vypočítá délku trasy, nastoupané a naklesané metry, sklon trasy mezi jednotlivými souřadnicemi, průměrný sklon celé trasy, maximální a průměrnou rychlost, jakou byla trasa projeta, a určí navigační body, které slouží pro navigaci po trase. Všechny tyto informace jsou ukládány do databáze. Aplikace také umožňuje vykreslení grafu výškového profilu trasy, grafu rychlosti v závislosti na ujeté vzdálenosti a grafu rychlosti v závislosti na čase jízdy. Všechny trasy je možné zobrazit na mapě s vyznačenou aktuální rychlostí, nadmořskou výškou nebo sklonem trasy v jednotlivých úsecích tratě. Na mapě lze také zobrazit navigační body, podle kterých probíhá navigace po trase. Jednotlivé trasy, a k nim uložené informace o jízdě, mohou být nahrány na informační systém, a naopak je zde možnost stáhnout trasy do mobilního zařízení, které jsou uloženy na informačním systému.

2.2 Popis funkcí mobilní aplikace

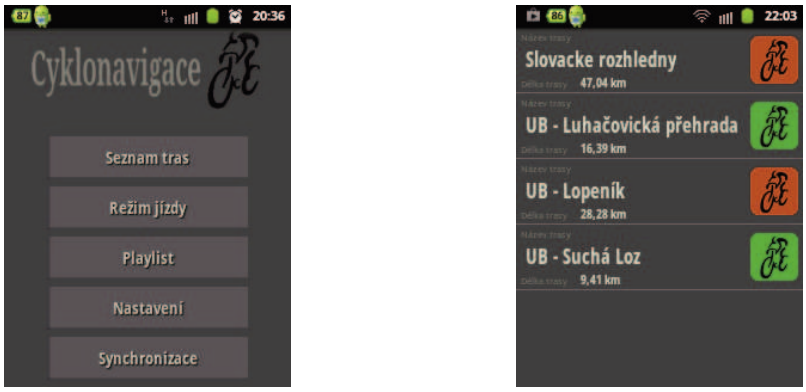
V této části se zaměřím na popis funkcí mobilní aplikace z uživatelského pohledu. Aplikace je určena pro platformu Android, k jejímu používání je potřeba mít k dispozici mobilní zařízení s tímto operačním systémem, minimálně ve verzi 2.2.1 Froyo a vyšší.

Po spuštění aplikace se zobrazí úvodní obrazovka (Obrázek 2.1.a). Aplikace je rozdělena do pěti částí (Seznam tras, Režim jízdy, Playlist, Nastavení a Synchronizace). Níže budou postupně popsány všechny funkce jednotlivých částí.

2.2.1 Seznam tras

Tato část aplikace bude sloužit pro procházení a manipulaci s trasami uloženými v databázi. V této části bude mít uživatel možnost zobrazit detail jednotlivých tras a detail informací o jízdě k nim uložených, včetně zobrazení grafu výškového profilu, grafu rychlosti v závislosti na ujeté vzdálenosti a grafu rychlosti v závislosti na čase jízdy a také zobra-

zit trasu na mapě. Dále zde bude mít uživatel možnost nahrávat trasy, a k nim uložené informace o jízdě, na informační systém.



(a) Úvodní obrazovka (b) Seznam tras

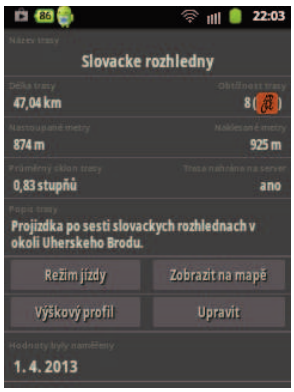
Obrázek 2.1: Úvodní obrazovka, Seznam tras

Při přesunutí do této části aplikace se uživateli zobrazí seznam všech tras uložených v databázi (Obrázek 2.1.b). U každé trasy bude zobrazen název, délka a obrázek cyklisty, reprezentující obtížnost trasy. Zelená barva znamená lehkou obtížnost, čím více bude obrázek zabarven do červena, tím je obtížnost trasy vyšší. Trasy bude možné seřadit podle několika kritérií, a to podle délky trasy, obtížnosti trasy, nastoupaných metrů a také podle naklesaných metrů. Možnost seřadit trasy bude v nabídce menu.

Po vybrání konkrétní trasy ze seznamu se uživateli zobrazí detail trasy (Obrázek 2.2.a). Na této obrazovce budou zobrazeny všechny informace o trase, její název, obtížnost (reprezentována obrázkem cyklisty a číselnou hodnotou), popis, délka, průměrný sklon, nastoupané a naklesané metry, dále zde budou tlačítka pro zapnutí režimu jízdy, pro zobrazení trasy na mapě, pro zobrazení grafu výškového profilu a pro úpravu základních informací o trase. Ve spodní části obrazovky bude zobrazen seznam informací o jízdě, které jsou uloženy k této trase. V nabídce menu budou položky pro úpravu a smazání trasy, a také položky sloužící pro synchronizaci s informačním systémem. Pokud nebude trasa nahrána na informační systém, bude zde položka sloužící pro nahrání trasy na informační systém, a pokud bude trasa nahrána na informační systém, budou zde položky pro stažení a nahrání souřadnic trasy na informační systém.

Detail informací o jízdě (Obrázek 2.2.b) se zobrazí po kliknutí na některou položku v seznamu informací o jízdě. V horní části obrazovky budou vypsány základní informace o trase, její název, délka a obtížnost. Dále zde budou zobrazeny informace naměřené při jízdě, den, kdy byly informace naměřeny, průměrná a maximální rychlost, čas jízdy,

průměrný sklon trasy a nastoupané a naklesané metry. Ve spodní části obrazovky budou tlačítka sloužící pro zobrazení grafu rychlosti v závislosti na ujeté vzdálenosti, grafu rychlosti a v závislosti na čase jízdy, grafu výškového profilu a také tlačítko sloužící pro zobrazení trasy, která byla při tomto měření zaznamenána. V nabídce menu bude položka sloužící pro smazání informací o jízdě, položka sloužící pro nahrání informací o jízdě na informační systém a také položka sloužící pro nahrání souřadnic ke trase. Souřadnice, které jsou uloženy k informacím o jízdě se mohou lišit od souřadnic, které jsou uloženy ke trase, proto zde bude možnost přepsání souřadnic trasy souřadnicemi uloženými k informacím o jízdě.



(a) Detail trasy

(b) Detail naměřených informací

Obrázek 2.2: Seznam tras, Detail trasy

Aplikace bude zobrazovat naměřené informace do tří grafů. Při zobrazení jakéhokoliv grafu bude přiblížení grafu nastaveno na hodnotu 1, to znamená, že na obrazovce bude vykreslen celý graf. Hlavně u delších tras bude graf značně nepřehledný, proto bude možné graf přiblížit, a to až 20x. Přiblížení grafu bude možné pomocí multitouch[1]. Pro posun přiblíženého grafu bude stačit přejet prstem na jednu či druhou stranu obrazovky.

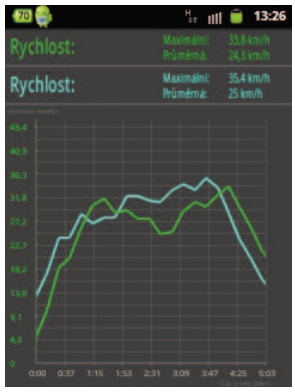
Graf výškového profilu trasy se bude mírně lišit od grafu výškového profilu informací o jízdě. V grafu výškového profilu informací o jízdě bude možné navíc zobrazit graf rychlosti v závislosti na ujeté vzdálenosti (Obrázek 2.3.a). V horní části obrazovky budou zobrazeny souhrnné informace o nadmořské výšce a v grafu výškového profilu informací o jízdě bude navíc zobrazena průměrná a maximální rychlost.

Do grafů rychlostí bude možné navíc zobrazit stejný graf, který je uložen k jiným

[1] multitouch - vícedotykové ovládání

informacím o jízdě, které jsou uloženy ke stejné trase (Obrázek 2.3.b). V horní části obrazovky bude zobrazena maximální a průměrná rychlost ke každému zobrazenému grafu.

(a) Graf výškového profilu

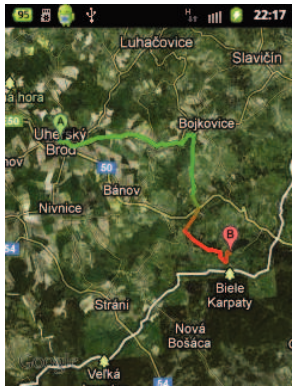


(b) Graf rychlosti v závislosti na čase jízdy

Obrázek 2.3: Grafy

Na mapě bude možné kromě samotné trasy zobrazit také navigační body, podle kterých probíhá navigace po trase (Obrázek 2.4.a). Možnost zobrazení navigačních bodů bude možné najít v nabídce menu. Budou se rozlišovat tři typy navigačních bodů, rovně, doprava a doleva. Trasu bude možné vykreslit i barevně, a to podle nadmořské výšky a sklonu trasy, a při zobrazení trasy informací o jízdě také podle aktuální rychlosti. Při zobrazení mapy podle nadmořské výšky a rychlosti bude nejvyšší hodnota reprezentována červenou barvou a nejnižší hodnota barvou zelenou (Obrázek 2.4.b).

(a) Zobrazení navigačních bodů



(b) Zobrazení trasy podle nadmořské výšky

Obrázek 2.4: Mapa

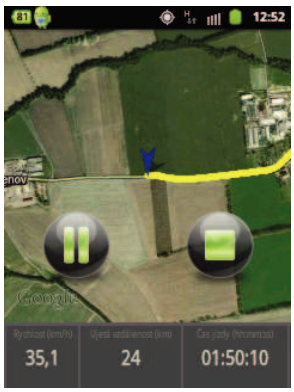
2.2.2 Režim jízdy

Tato část mobilní aplikace bude sloužit pro zaznamenávání informací o jízdě. Dále zde budou zobrazeny aktuální informace o jízdě, jako je aktuální rychlost, čas jízdy a ujetá vzdálenost, a také zde bude zobrazena mapa, na kterou bude vykreslována projížděná trasa.

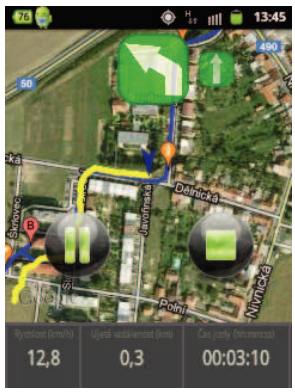
Režim jízdy bude možné zapnout ve dvou režimech. Normální režim bude sloužit pouze pro záznam informací o jízdě, a režim navigace bude sloužit, kromě záznamu informací o jízdě, také pro navigaci po trase. Při přechodu do režimu jízdy z úvodní obrazovky bude režim jízdy zapnut v normálním režimu, pro zapnutí režimu navigace bude nutné přejít do režimu jízdy z detailu nějaké trasy.

V normálním režimu se ve spodní části obrazovky budou vypisovat aktuální informace o jízdě a na mapě se bude postupně vykreslovat projížděná trasa. Dále zde budou tlačítka pro pozastavení a ukončení režimu jízdy (Obrázek 2.5.a).

V režimu navigace se spodní část obrazovky nebude lišit od normálního režimu. Na mapě bude navíc modrou barvou zobrazena trasa, po které bude navigace probíhat, a na ní navigační body, které ještě nejsou projety. Navíc bude ve vrchní části obrazovky panel, na kterém se bude zobrazovat směr aktuálního a následujícího navigačního bodu (Obrázek 2.5.b).



(a) Obrazovka v režimu jízdy



(b) Režim jízdy v navigačním režimu

Obrázek 2.5: Režim jízdy

Režim jízdy se bude spouštět tlačítkem *Start*, před samotným spuštěním se nejprve zjistí, jestli je zapnutý GPS přijímač. Pokud ne, režim jízdy se nespustí a na obrazovku se vypíše upozornění. Po spuštění režimu jízdy se začnou přehrávat skladby, které jsou

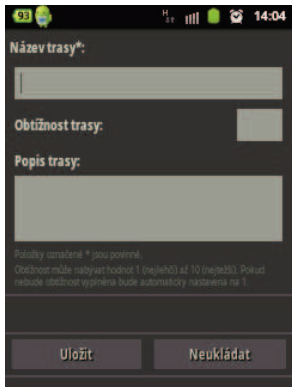
navoleny v sekci Playlist (viz níže) a začne připojování na GPS, uživatel bude hlasem informován o připojení GPS přijímače a také pokud GPS přijímač ztratí signál.

Pokud bude režim jízdy zapnut v režimu navigace, bude probíhat hlasová navigace po trase. Pokud bude dosažen navigační bod, přehraje se příslušný zvuk. Dále se bude kontrolovat, jestli je projížděná trasa stejná jako trasa, po které probíhá navigace. Pokud se uživatel vzdálí od trasy na více než 300 metrů, bude o tom hlasem informován. Pokud se do pěti minut nevrátí na trasu, bude režim jízdy přepnut do normálního režimu. Během tohoto odpočtu bude uživatel každou minutu hlasem informován, že se nachází mimo trasu. Uživatel bude hlasem informován i v případě, že se na trasu vrátí.

Režim jízdy bude možné ukončit tlačítkem *Stop*. Před samotným ukončením bude uživatel dotázán, jestli chce opravdu režim jízdy ukončit, jako ochrana před překliknutím. Po ukončení režimu jízdy se zobrazí obrazovka s naměřenými hodnotami (Obrázek 2.6.a). Před samotným uložením naměřených hodnot si uživatel bude moci prohlédnout graf rychlosti v závislosti na ujeté vzdálenosti, graf rychlosti v závislosti na čase jízdy a graf výškového profilu, bude moci také zobrazit projetou trasu na mapě, včetně zobrazení trasy podle nadmořské výšky, sklonu a rychlosti.

Naměřené hodnoty bude možné uložit kliknutím na tlačítko *Uložit*. Pokud bude režim jízdy zapnut v režimu navigace, budou naměřené informace uloženy ke trase, na kterou byl režim jízdy zapnut. Pokud bude režim jízdy zapnut v normálním režimu, zobrazí se uživateli obrazovka, na které bude vyzván k zadání názvu, obtížnosti a popisu trasy, a až poté budou naměřené informace uloženy do databáze.

(a) Obrazovka s naměřenými hodnotami



(b) Přidání nové trasy

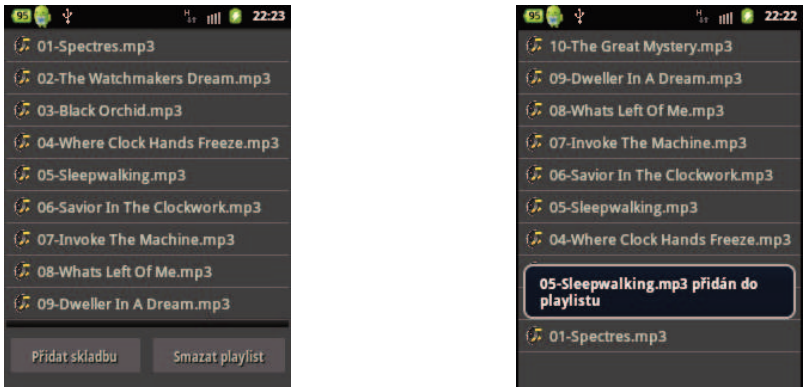
Obrázek 2.6: Naměřené informace

2.2.3 Playlist

Tato část aplikace bude sloužit pro vybrání skladeb, které budou přehrávány v režimu jízdy. Do playlistu bude možné přidat jakoukoliv skladbu ve formátu mp3, která je uložena na SD kartě.

Při přechodu do této sekce se nejprve zobrazí seznam navolených skladeb (Obrázek 2.7.a). Ve spodní části obrazovky budou tlačítka pro přidání skladby do playlistu a pro smazání celého playlistu. Jednotlivé skladby bude možné z playlistu smazat kliknutím na název skladby.

Přidání skladby do playlistu bude probíhat následovným způsobem. Po kliknutí na tlačítko *Přidat skladbu* se zobrazí domovský adresář SD karty, budou se zde zobrazovat pouze adresáře a soubory ve formátu mp3. Kliknutím na adresář se zobrazí adresáře a mp3 soubory obsažené v daném adresáři, o úroveň výš bude možné jít tlačítkem zpět. Kliknutím na mp3 soubor se vybraná skladba přidá do playlistu a zobrazí se upozornění o přidání dané skladby do playlistu (Obrázek 2.7.b).



(a) Obrazovka playlistu (b) Procházení složek

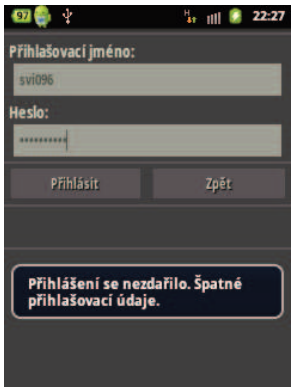
Obrázek 2.7: Playlist

2.2.4 Nastavení

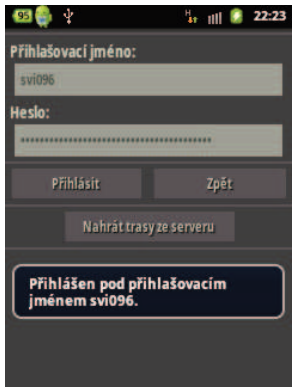
Tato část aplikace bude sloužit pro nastavení vlastností aplikace. Bude zde možné nastavit typ zobrazované mapy, dále zde bude možné určit, zda má display v režimu jízdy zůstat po celou dobu jízdy rozsvícený. A také zde bude možnost vymazání celé databáze. Před vymazáním databáze bude uživatel vyzván k potvrzení.

2.2.5 Synchronizace

Tato část aplikace bude sloužit pro synchronizaci tras a informací o jízdě s informačním systémem. Při přechodu do této části aplikace se nejprve provede ověření přihlašovacího jména a hesla, pokud budou tyto údaje uloženy v databázi. Pokud ne, uživatel bude vyzván k zadání těchto údajů. Pokud ověření přihlašovacích údajů z nějakého důvodu selže, zobrazí se upozornění s důvodem selhání (Obrázek 2.8.a). Po úspěšném ověření přihlašovacích údajů, proběhne synchronizace tras a informací o jízdě. Zjistí se, které trasy a informace o jízdě jsou nahrány na informačním systému a zobrazí se tlačítko *Nahrát trasy ze serveru* (Obrázek 2.8.b). Po kliknutí na toto tlačítko se stáhnou všechny trasy uživatele, které čekají na nahrání na mobilní zařízení (viz. kapitola Popis funkcí informačního systému).



(a) Upozornění o selhání přihlášení



(b) Vzhled obrazovky po úspěšném přihlášení

Obrázek 2.8: Synchronizace

3 Implementace mobilní aplikace

V této kapitole budou popsány hlavní funkce mobilní aplikace z programátorského pohledu. Budou zde zjednodušeně popsány důležité funkce mobilní aplikace, jako je navigace po trase a sledování odchýlení od trasy, práce s databází, práce se službami, algoritmus pro výpočet navigačních bodů a volání WCF služeb.

3.1 Struktura mobilní aplikace

Aplikace má klasickou adresářovou strukturu, typickou pro Android Project. Dále bude popsán obsah jednolivých adresářů, s vyjímkou automaticky generovaných adresářů `bin` a `gen`.

- **`res`** - Tento adresář slouží pro ukládání zdrojů použitých v aplikaci.
  - **`drawable-hdpi`** - V tomto adresáři jsou uloženy obrázky, které jsou v aplikaci použity včetně ikon zobrazených v menu, markerů zobrazených na mapě a obrázků zobrazovaných při navigaci po trase. Také jsou zde uloženy XML soubory, které definují vzhled jednotlivých grafických prvků aplikace.
  - **`layout`** - Tento adrešář obsahuje XML soubory definující vzhled jednotlivých aktivit.
  - **`menu`** - Tento adrešář obsahuje XML soubory definující položky jednotlivých menu použitých v aplikaci.
  - **`raw`** - V tomto adresáři jsou uloženy zvuky, které jsou přehrávány v režimu jízdy. Nejsou zde uloženy skladby, které jsou přidány do playlistu. Zvuky jsou uloženy ve formátu mp3.
  - **`values`** - Tento adresář obsahuje XML soubory, ve kterých je uložena drtivá většina textů použitých v aplikaci. Dále je zde XML soubor, který přiřazuje k jednotlivým grafickým prvkům nový vzhled, který je definován v adresáři `drawable-hdpi`.
- **`src`** - V tomto adresáři jsou uloženy zdrojové kódy aplikace. Pro přehlednost jsou jednotlivé třídy rozděleny do 7 balíčků.
  - **`additionalClass`** - Tento balíček obsahuje pomocné třídy sloužící pro výpočet navigačních bodů, a také třídy sloužící pro vykreslování trasy na mapě.
  - **`models`** - Tento balíček obsahuje pro každou tabulku v databázi jednu třídu, která reprezentuje datovou strukturu této tabulky.

  - **DAO** - Tento balíček tříd obsahuje pro každou tabulku v databázi jednu třídu, pomocí které se manipuluje s daty uloženými v databázi.
  - **overlays** - Tento balíček obsahuje třídy, které reprezentují objekty vykreslované na mapě.
  - **services** - V tomto balíčku jsou uloženy služby. Je zde služba `DatabaseManager.java`, která se stará o práci s databází a volání WCF služeb, a třída `RideService.java`, která zaznamenává informace o jízdě.
  - **WCF** - Tento balíček obsahuje třídy, které slouží pro volání WCF služeb.
  - **gui** - V tomto balíčku jsou uloženy aktivity, které jsou použity v aplikaci.

- **AndroidManifest.xml** - V tomto souboru jsou definovány jednotlivé služby a aktivity. Dále se zde nastavují práva aplikace přístupu k zabezpečeným částem aplikačního frameworku.

3.2 Služby (Service)

Android služby představují proces běžící na pozadí. Služby[1] neposkytují uživatelské rozhraní, používají se k vykonávání dlouho trvajících úkolů, nebo k přístupu ke vzdáleným zdrojům, kde není známá doba odezvy. Služba může být spuštěna z aktivity. Aby bylo možné volat metody implementované ve službě z aktivity, je nejprve nutné tuto službu k aktivitě připojit. Jednu službu je možné připojit k více aktivitám.

V aplikaci jsou použity dvě služby. Služba `RideService.java` slouží pro záznam informací o jízdě, navigaci po trase, sledování odchýlení od trasy, výpočet navigačních bodů a uložení naměřených informací o jízdě do databáze. Pro manipulaci s daty uloženými v databázi, a pro volání WCF služeb slouží služba `DatabaseManager.java`.

V každé aktivitě, ze které jsou volány metody implementovány ve službě, musí být definován Handler, který je spuštěn zároveň se službou. Tento Handler slouží pro volání metod ze služby. Dále je zde definováno rozhraní Runnable[2], ve kterém je definováno jádro Handleru. V Handleru je volána metoda `checkService()`, ve které jsou volány metody implementované ve službě (Výpis 3.1).

V Handleru je definována metoda `handleMessage(Message msg)`, která je volána vždy, když je handleru poslána nějaká zpráva. V této metodě je volána metoda `checkService()`, pomocí které jsou volány metody implementované ve službě. V metodě `checkService()` se nejprve ověří, zda je služba spuštěna a připojena k aktivitě, poté jsou podle zvolené operace volány metody služby, po vykonání těchto metod se zastaví Timer, který běží v Handleru, a zruší se služba. Rozhraní Runnable definuje jádro

Handleru, je zde spuštěn Timer, který každých 100 milisekund pošle prázdnou zprávu Handleru, tím je zavolána metoda `handleMessage(Message msg)`.

```
private Handler mHandler = new Handler() {
    public void handleMessage(Message msg) {
        checkService();
}};
protected void checkService() {
    if (isBound && isRun) {
        if (operation == OPERATION_GETDRIVINGIINFORMATION) {
        //podle zvolené operace se zavolají požadované metody ze služby
            if (timer != null)
          timer.cancel();
            this.stopService();
}}}
private final Runnable runTimer = new Runnable() {
  public void run() {
    timer = new Timer();
    timer.scheduleAtFixedRate(
      new TimerTask () {
       public void run(){
         mHandler.sendEmptyMessage(0);
       }}, 0, 100);
}};
```

Výpis 3.1: Definice Handleru a rozhraní Runnable

Dále jsou v každé aktivitě, která volá metody ze služby, implementovány dvě metody. Metoda pro připojení služby k aktivitě `doBindService()` a metoda pro odpojení služby od aktivity `doUnbindService()` (Výpis 3.2). Služba je od aktivity odpojena vždy, když je aktivita přesunuta do pozadí, tzn. zhasne obrazovka nebo je zobrazena jiná aktivita, a opětovně připojena, pokud je aktivita přesunuta do popředí, rozsvití-li se display nebo je aktivita znovu zobrazena.

V metodě `doBindService()` se nejprve nastaví proměnná, která určuje, zda je služba připojena k aktivitě, na true, poté je služba pomocí metody `bindService(Intent, ServiceConnection, int)` připojena k aktivitě, a je spušten Handler.

V metodě `doUnbindService()` se nejprve nastaví proměná, která určuje zda je služba připojena k aktivitě, na false, poté je služba od aktivity odpojena pomocí metody `unBindService(ServiceConnection)`, a nakonec je zastaven Timer, který běží v těle Handleru.

```
protected void doBindService() {
  isBound = true;
  bindService(new Intent(RouteDetail.this, DatabaseManager.class), mConnection, Context.
      BIND_AUTO_CREATE);
  mHandler.postDelayed(runTimer, 100);
}
protected void doUnbindService() {
  isBound = false;
  unbindService(mConnection);
  if (timer != null)
    timer.cancel();
}
```

Výpis 3.2: Připojení a odpojení služby od aktivity

V poslední řadě jsou v každé aktivitě pracující se službou implementovány metody pro spuštění a zrušení služby. Pro spuštění je implementována metoda `startService()`, a pro zrušení služby je implementována metoda `stopService()` (Výpis 3.3).

V metodě `startService()` se nejdříve ověří, zda není služba spuštěna, poté je služba spuštěna pomocí metody `startService(Intent)`. Nakonec je do proměnné, určující zda služba běží, přiřazena hodnota true, a služba se připojí k aktivitě.

V metodě `stopService()` je nejprve služba odpojena od aktivity, poté je služba zrušena pomocí metody `stopService(Intent)` a nakonec je do proměnné, určující zda služba běží, přiřazena hodnota false.

```
protected void startService() {
    if (!isRun) {
        Intent intent = new Intent (this, DatabaseManager.class);
        startService(intent);
        isRun = true;
        doBindService();
    }
}
protected void stopService() {
    doUnbindService();
    stopService(new Intent(this, DatabaseManager.class));
    isRun = false;
}
```

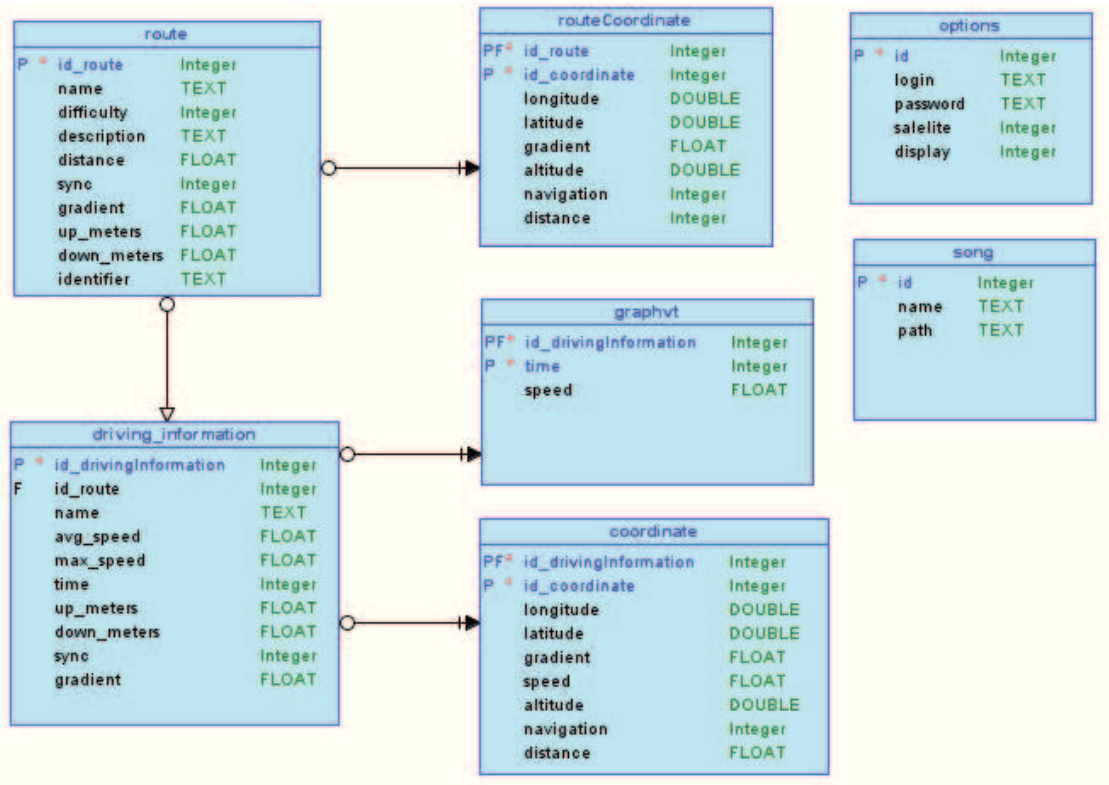
Výpis 3.3: Spuštění a zrušení služby

3.3 Databáze

Android má integrován SQLite[3] vytváření databází. Manipulace s daty probíhá pomocí SQL příkazů. Veškerá manipulace s daty, uloženými v databázi, probíhá ve službě `DatabaseManager.java`.

3.3.1 Struktura databáze

Databáze obsahuje celkem sedm tabulek (Obrázek 3.1). Pět z nich slouží pro ukládání tras a informací o jízdě, jedna tabulka slouží pro ukládání skladeb, které jsou přidány do playlistu, a poslední tabulka slouží pro uložení nastavení aplikace.



Obrázek 3.1: Schéma databáze mobilní aplikace

- **route** - Tabulka, do které se ukládají základní informace o trase. Ke každé trase se ukládá ID, název, obtížnost, popis, identifikační značka, která je pro každou trasu unikátní a slouží pro synchronizaci tras s informačním systémem, nastoupané metry, naklesané metry, průměrný sklon trasy a proměnná, která určuje jestli je trasa nahrána na informační systém či nikoliv.

- **routeCoordinates** - Tabulka, do které se ukládají souřadnice, které jsou naměřeny k jednotlivým trasám. Ke každé souřadnici se ukládá ID trasy, ke které je souřadnice uložena, ID souřadnice, které určuje pořadí souřadnic v jakém jsou naměřeny, u každé trasy jsou souřadnice číslovány od 1, zeměpisná délka (longitude), zeměpisná šířka (latitude), sklon, nadmořská výška, vzdálenost od počátku trasy a navigační značka souřadnice.
- **drivingInformation** - Tabulka, do které se ukládají hodnoty k informacím o jízdě. Ukládá se zde ID, název, což je datum, kdy byly hodnoty naměřeny, ID trasy, ke které jsou informace o jízdě uloženy, průměrná rychlost, maximální rychlost, čas jízdy, nastoupané metry, naklesané metry, průměrný sklon, a proměnná, která určuje, jestli jsou informace o jízdě nahrány na informační systém či nikoliv.
- **coordinates** -Tabulka, do které se ukládají souřadnice, které jsou uloženy k jednotlivým informacím o jízdě. Ke každé souřadnici se ukládá ID informací o jízdě, ID souřadnice, zeměpisná délka (longitude), zeměpisná šířka (latitude), sklon, nadmořská výška, vzdálenost od počátku trasy, rychlost a navigační značka souřadnice.
- **graphVT** - Tabulka, do které se ukládají body, potřebné pro vykreslení grafu rychlosti v závislosti na čase jízdy. Ke každému bodu se ukládá ID informací o jízdě, ke kterým je bod uložen, čas a rychlost.
- **song** - Tabulka, do které se ukládají skladby, které jsou v playlistu. U každé skladby je uloženo ID, její název a cesta, kde je skladba uložena.
- **options** - Tabulka, do které se ukládá nastavení aplikace. Je zde uloženo přihlašovací jméno, přihlašovací heslo, hashované pomocí SHA1, a dále také proměnná určující, zda má být zobrazovaná satelitní či klasická mapa, a proměnná určující, zda má obrazovka v režimu jízdy zhasínat či nikoliv.

3.3.2 Přístup k datům v databázi

Každá tabulka má definované dvě třídy, jedna třída, z balíčku `models`, reprezentuje datovou strukturu dané tabulky, a ve druhé třídě, z balíčku `DAO`, jsou definovány všechny operace prováděné nad konkrétní tabulkou. Jakákoliv manipulace s databází je prováděna ve službě `DatabaseManager.java`, z této služby jsou volány metody definované pro operace select, insert, update a delete z tříd uložených v balíčku `DAO` (Výpis 3.4).

Metoda `open()` vytvoří instanci třídy `SQLiteDatabase`, která slouží pro přístup do databáze, metoda `close()` zavře databázi a metoda `addRoute(Route)` nejprve otevře databázi, poté zavolá metodu pro vložení nové trasy do databáze, ze třídy `RouteDAO`, ve které jsou implemetovány metody pro práci s daty v tabulce Route, nakonec databázi zavře a vrátí ID vložené trasy.

```
private SQLiteDatabase db;
private SQLiteHelper helper = new SQLiteHelper(this);

public void open() throws SQLException {
  this.db = helper.getWritableDatabase();
  //služba
}
public void close() {
   helper.close();
}
public int addRoute (Route route) {
   this.open();
   int output = RouteDAO.addRoute(db, route);
   this.close();
   return output;
}
```

Výpis 3.4: Ukázka volání metod ze služby

Z důvodu časové náročnosti ukládání většího počtu dat do databáze, jsou pro vkládání souřadnic a bodů grafu rychlosti v závislosti na čase jízdy použity transakce a hromadné operace (Výpis 3.5).

Metoda `saveGraphVT(ArrayList<GraphVT>, SQLiteDatabase)` ukládá do databáze body grafu rychlosti v závislosti na čase jízdy. Nejprve se vytvoří nová transakce a SQL příkaz pro vložení jednoho záznamu do tabulky `graphVT`, dále je vytvořena instance třídy `SQLiteStatement`[4], pomocí které jsou vkládány jednotlivé záznamy do databáze. Poté jsou do databáze postupně vloženy všechny body grafu. Pokud se při vkládání jednotlivých bodů nevyskytne žádná chyba, transakce je označena za úspěšnou a ukončena, tzn. je proveden *commit* a všechny změny jsou uloženy do databáze. Pokud se při vkládání bodů vyskytne nějaká chyba, jsou informace o chybě vypsány do logu a transakce se ukončí, jelikož tato transakce nebyla označena jako úspěšná, provede se *rollback* a databáze se vrátí do stavu v jakém byla před začátkem transakce.

```
public static void saveGraphVT(ArrayList<GraphVT> points, SQLiteDatabase db) {
    db.beginTransaction();
    try {
        String sql = //vytvoření SQL příkazu
        SQLiteStatement insert = db.compileStatement(sql);
        for (int i = 0; i < points.size(); i++) {
            //navázání ukládáných hodnot kontrétního bodu na proměnné v SQL příkazu
            insert.executeInsert();
        }
        db.setTransactionSuccessful();
    } catch (Exception e) {
      //vypsání informací o chybě do logu
    } finally {
        db.endTransaction();
    }
}
```

Výpis 3.5: Ukázka použití transakcí a hromadných operací

3.4 Režim jízdy

Pro režim jízdy je implementována aktivita `Ride.java` a služba `RideService.java`. Načítání dat z GPS přijímače a navigace po trase probíhá ve službě. Aktivita, pokud je na popředí, si každou sekundu od služby zjišťuje aktuální informace o jízdě a tyto informace zobrazí na obrazovce.

V režimu jízdy probíhá značné množství operací, proto jsem popis jednotlivých funkcí rozdělil do několika částí. Postupně budou popsány funkce Timerů, Location Listenerů a GpsStatus Listeneru, které jsou při jízdě spuštěny. A dále bude popsán princip kontroly projížděné trasy a princip navigace po trase.

3.4.1 Timery

V režimu jízdy jsou implementovány celkem tři Timery[5].

Timer `timer` slouží pro počítání času jízdy a pro uložení bodů grafu rychlosti v závislosti na čase jízdy, tyto body jsou ukládány v 15ti sekundových intervalech. Timer tiká každou sekundu, každé tiknutí zvýší hodnotu proměnné, ve které je uložen čas jízdy, o hodnotu 1, a pokud je čas dělitelný patnácti beze zbytku, uloží se bod grafu rychlosti v závislosti na čase jízdy. Tento Timer je spuštěn při zapnutí režimu jízdy.

Timer `playTimer` posílá dvakrát za sekundu prázdnou zprávu Handleru, který se stará o přehrávání zvuků (viz. sekce Přehrávání zvuků). Tento Timer je také spuštěn při zapnutí režimu jízdy.

Timer `outTimer` je spuštěn vždy, když se uživatel vzdálí na více než 300 metrů od trasy, po které probíhá navigace. Tento Timer tiká jednou za minutu a při každém tiknutí přehraje zvuk, který uživateli oznámí čas, který mu zbývá pro návrat na trasu. Timer je ukončen, pokud se uživatel vrátí na trasu, a nebo po uplynutí pěti minut, kdy je režim jízdy přepnut do normálního režimu.

3.4.2 Přehrávání zvuků

O přehrávání zvuků se stará Handler `playHandler`, který po přijmutí zprávy zavolá metodu `checkPlaying()`. Handleru posílá dvakrát za sekundu prázdnou zprávu Timer `playTimer`.

Pro přehrávání zvuků slouží dva MediaPlayery[6] (Výpis 3.6). MediaPlayer `mp3` slouží pro přehrávání skladeb, které jsou navoleny v sekci Playlist, a MediaPlayer `sound` slouží pro přehrávání zvuků sloužících k navigaci po trase a pro přehrávání oznamovacích zvuků, jako je spuštění a pozastavení režimu jízdy, připojení GPS, a ztráta GPS signálu.

```
if (mp3.isPlaying()) {
  if (playNavigation != Sounds.NONE && !sound.isPlaying()) {
    AssetFileDescriptor afd = getResources().openRawResourceFd(playNavigation);
    mp3.setVolume(0.15f, 0.15f); volumeDown = true;
    sound.reset();
    sound.setDataSource(afd.getFileDescriptor(), afd.getStartOffset(), afd.getDeclaredLength());
    sound.prepare(); sound.start(); afd.close();
    playNavigation = Sounds.NONE;
  }
  if (volumeDown && !sound.isPlaying()) {
    mp3.setVolume(1f, 1f);
    volumeDown = false;
  }
} else {
    mp3.reset();
    mp3.setDataSource(songs.get(actualSong).path);
    mp3.prepare(); mp3.start();
    actualSong ++;
    if (actualSong == songs.size())
      actualSong = 0;
}
```

Výpis 3.6: Ukázka přehrávání zvuků v režimu jízdy

Do proměnné `playNavigation` se ukládá ID zvuku, který má být přehrán, v poli `songs` jsou uloženy skladby, které jsou navoleny v sekci Playlist. V proměnné `actualSong` je uloženo ID právě přehrávané skladby z pole `songs`.

Nejdříve se ověřuje, zda MediaPlayer `mp3` hraje. Pokud ano, zjišťuje se jestli je v proměnné `playNavigation` uložen zvuk, který se má přehrát, a zároveň jestli MediaPlayer `sound` nehraje, tím je zajištěno, že se nepřehraje více zvuků najednou. Pokud jsou splněny obě podmínky, vytvoří se instance třídy `AssetFileDescriptor`, pomocí které se zjistí cesta k souboru, ve kterém je uložen zvuk, který se má přehrát. Poté je ztlumen MediaPlayer `mp3`, aby byl přehrávaný zvuk zřetelně slyšet a do proměnné `volumeDown` se přiřadí hodnota true. Dále je MediaPlayer `sound` resetován a nastaví se cesta k souboru, ve kterém je uložen zvuk, který se má přehrát. Poté je MediaPlayer `sound` spuštěn, `AssetFileDescriptor` se zavře a do proměnné `playNavigation` se přiřadí hodnota, která nereprezentuje žádný zvuk. Pokud je MediaPlayer `mp3` ztlumen a MediaPlayer `sound` už dohrál, nastaví se hlasitost MediaPlayeru `mp3` na maximum, a do proměnné `volumeDown` se přiřadí hodnota false.

Pokud MediaPlayer `mp3` nehraje, začne se přehrávat další skladba, která je uložena v poli `songs`. Pokud jsou přehrány všechny skladby, které jsou přidány do playlistu, přehrávání pokračuje opět od první skladby.

3.4.3 LocationListenery

V režimu jízdy jsou spuštěny dva LocationListenery[7]. Pomocí LocationListeneru `CheckJoinGpsListener` se zaznamenává aktuální rychlost, a také se pomocí něj určuje, zda GPS přijímač přijímá data či nikoliv. Tento LocationListener načítá data z GPS přijímače každé dvě sekundy, ukládá se zde pouze aktuální rychlost a také přesný čas načtení dat z GPS přijímače.

LocationListener `CoordinatesListener` slouží pro načítání souřadnic projížděné trasy a pro výpočet ujeté vzdálenosti. Pokud je režim jízdy v navigačním režimu, je zde volána metoda, která kontroluje odchýlení od trasy, po které probíhá navigace, a metoda pomocí které probíhá navigace po trase. Tento LocationListener načítá data z GPS přijímače, pokud je vzdálenost, mezi aktuální polohou a polohou poslední naměřené souřadnice, větší než pět metrů.

V navigačním i normálním režimu probíhá v `CoordinatesListeneru` výpočet ujeté vzdálenosti a uložení aktuální pozice do pole `locations` (Výpis 3.7).

Do proměnné `preview` je ukládána poslední načtená poloha a do proměnné

`distance` se ukládá ujetá vzdálenost. Pokud není v `preview` uložena žádná souřadnice, načtená poloha je první souřadnice trasy, proto se ujetá vzdálenost nastaví na nulu. Pokud je v `preview` uložena souřadnice, přičte se k ujeté vzdálenosti, vzdálenost mezi aktuální polohou a poslední načtenou polohou. Nakonec se aktuální poloha uloží do pole `locations` a také do proměnné `preview`.

```
if (preview == null) {
    distance = 0;
} else {
    float dist = location.distanceTo(preview);
    distance += dist;
}
locations.add(location);
preview = location;
```

Výpis 3.7: Výpočet ujeté vzdálenosti a uložení aktuální polohy

Pokud je režim jízdy zapnut v navigačním režimu, je v `CoordinatesListeneru` také kontrolováno odchýlení od projížděné trasy a probíhá zde navigace po trase (Výpis 3.8).

```
if (mode == Constants.RIDE_MODE_NAVIGATION) {
  if (!runOutTimer) {
    if (preview == null)
      tempIndex = NOTFIND;
    if (tempIndex == FIND) {
      checkRoute(location);
      checkLocation(location);
    } else {
      if (index_check != -1)
        checkRoute(location);
      actualPoint = findFirstNavigationPoint(location);
      if (actualPoint != -1) {
        index_check = getImmediateCheckPoint(location);
        tempIndex = FIND; }
    }
  } else {
    if (countMinutes == 6) {
      mode = Constants.RIDE_MODE_NORMAL;
      //vypnutí outTimeru a přehrání zvuku o přepnutí režimu jízdy do normálního režimu
    } else {
      checkRoute(location);
      tempIndex = NOTFIND;
}}}
```

Výpis 3.8: Navigace po trase a sledování odchýlení od trasy

Proměnná `tempIndex` reprezentuje, zda byl nalezen první navigační bod či nikoliv. Projížděná trasa nemusí být projížděna od začátku, proto se nejprve hledá první navigační bod, a až poté probíhá navigace po trase. V proměnné `index_check` je uložen index aktuálního kotrolního bodu, podle kterých probíhá sledování odchýlení od projížděné trasy. V proměnné `actualPoint` je uložen index následujícího navigačního bodu. Metoda `checkLocation(Location)` se stará o navigaci po trase, tato metoda bude podrobněji popsána v sekci Navigace po trase. Metoda `checkRoute(Location)` kontroluje odchýlení od projížděné trasy, tato metoda bude podrobněji popsána v sekci Kontrola projížděné trasy. Metoda `getImmediateCheckPoint(Location)` vrací index nejbližšího kotrolního bodu. Metoda `findFirstNavigationPoint(Location)` prochází všechny navigační body a pokud je aktuální poloha v blízkosti některého navigačního bodu, přehraje zvuk a vrátí index následujícího navigačního bodu.

Nejprve se zjistí, zda není spušten Timer, který signalizuje, že se uživatel nachází mimo trasu, po které probíhá navigace. Pokud není, zjistí se, zda naměřená poloha není první souřadnicí projížděné trasy. Pokud ano, nastaví se do proměnné `tempIndex` hodnota `NOTFIND`, tzn. že ještě nebyl nalezen první navigační bod. Poté se zjistí, zda je nalezen první navigační bod či nikoliv. Pokud je tento navigační bod nalezen, ověří se odchýlení od trasy pomocí metody `checkRoute(Location)` a proběhne navigace po trase pomocí metody `checkLocation(Location)`. Pokud není nalezen první navigační bod, ověří se, zda je nalezen index nejbližšího kontrolního bodu. Pokud ano, zavolá se metoda `checkRoute(Location)` pro ověření odchylky od projížděné trasy. K tomuto ověření odchýlení od trasy dojde pouze v případě, že se uživatel odchýlí od trasy a poté na trasu vrátí. Dále se zjišťuje index prvního navigačního bodu pomocí metody `findFirstNavigationPoint(Location)`. Pokud je nalezen, zjistí se index nejbližšího kontrolního bodu a do proměnné `tempIndex` je přiřazena hodnota `FIND`, která reprezentuje, že byl nalezen první navigační bod. Pokud je spuštěn Timer, který signalizuje, že se uživatel nachází mimo trasu. Zjistí se, zda není spuštěn více než pět minut. Pokud ano, přepne se režim jízdy do normálního režimu. Pokud ne, kontroluje se, zda se uživatel nevrátil na trasu pomocí metody `checkRoute(Location)` a do proměnné `tempIndex` se přiřadí hodnota `NOTFIND`.

3.4.4 GpsStatus Listener

V režimu jízdy je spuštěn jeden GpsStatus Listener[8] `CheckJoinGpsStatus`. Pomocí něj probíhá kontrola, zda je připojen GPS přijímač či nikoliv. Pokaždé, když jsou načteny data z GPS přijímače, je zavolána metoda `onGpsStatusChanged(int)` (Výpis 3.9). V této metodě se zjišťuje čas od posledního načtení dat z GPS přijímače a podle tohoto času

se určuje zda je GPS přijímač připojen či nikoliv.

V proměnné `timePreviewLocationLoad` je uložen čas posledního načtení souřadnic v `CheckJoinGpsListeneru`, který načítá data z GPS přijímače každé dvě sekundy. Proměnné `playsJoin` a `playsLost` jsou pomocné proměnné, které zajišťují, aby se zvuk ztráty GPS signálu a zvuk připojení GPS přijímače přehrál pouze jednou.

```
if (preview != null)
   joinGPS = (SystemClock.elapsedRealtime() − timePreviewLocationLoad) < 4000;
   if (joinGPS) {
      if (!playsJoin) {
         playNavigation = AdditionalMethods.getSound(Sounds.JOIN);
         playsJoin = true; playsLost = false;
   }}
   else {
      if (!playsLost) {
         playNavigation = AdditionalMethods.getSound(Sounds.LOST);
         tempIndex = NOTFIND;
         playsJoin = false; playsLost = true;
}}
```

Výpis 3.9: Kontrola připojení GPS přijímače

Nejprve se do proměnné `joinGPS` přiřadí hodnota true nebo false, podle toho jestli je čas od posledního načtení dat z GPS přijímače menší než čtyři sekundy či nikoliv. A poté se podle této proměnné přehraje zvuk informující o připojení GPS přijímače či ztrátě GPS signálu, a do pomocných proměnných `playsJoin` a `playsLost` jsou přiřazeny hodnoty, tak aby nebyl příslušný zvuk přehrán více než jednou.

3.4.5 Kontrola projížděné trasy

Kontrola projížděné trasy probíhá pomocí kontrolních bodů. Pokud je režim jízdy zapnut v navigačním režimu, jsou po celé trase, po které probíhá navigace, každých 200 metrů vytvořeny kontrolní body. Kontrola projížděné trasy probíhá tak, že se zjišťují vzdálenosti aktuální polohy od nejbližších dvou kontrolních bodů, a podle těchto vzdáleností se určuje, zda je uživatel na trase či nikoliv (Výpis 3.10).

Nejprve se zjistí, zda není spuštěn Timer, který signalizuje, že se uživatel nachází mimo trasu, po které probíhá navigace.

Pokud Timer spuštěn není, uživatel se nachází na trase a ověřuje se, zda je tomu tak i nadále. Nejdříve se zjistí, zda není aktuální kontrolní bod poslední. Pokud není, do proměnné `distance_1` se uloží vzdálenost aktuální polohy od aktuálního kontrolního

bodu, a do proměnné `distance_2` se uloží vzdálenost aktuální polohy od následujícího kontrolního bodu. Pokud je vzdálenost od aktuálního kontrolního bodu menší než 300 metrů, uživatel je stále na trase. Pokud je tato vzdálenost větší než 300 metrů, ověří se, zda je vzdálenost od následujícího kontrolního bodu menší než 300 metrů. Pokud ano, nastaví se tento kontrolní bod jako aktuální. Pokud ne, uživatel se nachází mimo trasu a je spuštěn Timer, který signalizuje, že se uživatel nachází mimo trasu.

Pokud je Timer spuštěn, uživatel se nachází mimo trasu a probíhá kontrola, zda se uživatel nevrátil na trasu. To znamená, že se prochází všechny kontrolní body a zjišťuje se, zda jsou vzdálenosti, mezi aktuální polohou a dvěma po sobě následujícími kontrolními body, menší než 300 metrů. Pokud se takové kontrolní body najdou, uživatel se vrátil na trasu. Do proměnné `index_check` je uložen index nejbližšího kontrolního bodu a také je vypnut Timer, který signalizuje, že se uživatel nachází mimo trasu.

```
if (!runOutTimer) {
  if ((index_check + 1) < checkPoints.size()) {
    float distance_1 = loc.distanceTo(checkPoints.get(index_check));
    float distance_2 = loc.distanceTo(checkPoints.get(index_check + 1));
    if (distance_1 > 300) {
      if (distance_2 < 300)
        index_check ++;
        else {
        runOutTimer = true;
        //zapnutí Timeru, který signalizuje, že se uživatel odchýlil od trasy
  }}}
} else {
  for (int i = 0; i < checkPoints.size() - 1; i++) {
    float distance_1 = loc.distanceTo(checkPoints.get(i));
    float distance_2 = loc.distanceTo(checkPoints.get(i + 1));
    if (distance_1 < 300 && distance_2 < 300) {
      playNavigation = AdditionalMethods.getSound(Sounds.BACK);
      index_check = i;
      //vypnutí Timeru, který signalizuje, že se uživatel odchýlil od trasy
      runOutTimer = false; break;
}}}
```

Výpis 3.10: Kontrola projížděné trasy

3.4.6 Navigace po trase

O navigaci po trase se stará metoda `checkLocation(Location)`. Zvuk navigace je přehrán vždy, když je aktuální poloha v určité vzdálenosti od navigačního bodu a zá-

roveň dostatečně daleko od předchozího navigačního bodu. Vzdálenost, na kterou se přehrává zvuk navigace, je určena podle aktuální rychlosti (Tabulka 3.1).

Aktuální rychlost	Vzdálenost od aktuálního	Vzdálenost od předchozího
0 - 20 km/h	20 metrů	15 metrů
20 - 40 km/h	40 metrů	35 metrů
40-60 km/h	60 metrů	55 metrů
60-80 km/h	80 metrů	75 metrů
80-100 km/h	100 metrů	95 metrů
> 100 km/h	200 metrů	195 metrů

Tabulka 3.1: Tabulka vzdálenosti pro přehrávání navigačních zvuků

Nejdříve se hledá první navigační bod, toto hledání probíhá obdobně jako navigace po trase, ale prochází se všechny navigační body a nekontroluje se vzdálenost od předchozího navigačního bodu. Pokud je nalezen první navigační bod, začne probíhat navigace po trase pomocí metody `checkLocation(Location)` (Výpis 3.11).

```
int indexFrom = actualPoint;
int indexTo = actualPoint + 20;
//ověření zda nejsou indexy mimo rozsah pole
for (int i = indexFrom; i <= indexTo; i++) {
  float temp_distance = 200;
  if (i - 1 > 0)
    temp_distance = loc.distanceTo(navigationPoints.get(i - 1).location);
  float distance = loc.distanceTo(navigationPoints.get(i).location);
  int range = 0;
  //nastavení range podle aktuální rychlosti
  if (distance <= range && temp_distance >= range - 5) {
    playNavigation = AdditionalMethods.getSound(navigationPoints.get(i).sound);
    if (navigationPoints.get(i).sound != Sounds.END) {
      actualPoint = i + 1; break;
    } else {
      end = true; break;
}}}
```

Výpis 3.11: Ukázka navigace po trase

Při navigaci se testuje následujících dvacet navigačních bodů od aktuálního navigačního bodu. Nejdříve jsou určeny indexy navigačních bodů, které se budou testovat. Poté jsou tyto navigační body postupně procházeny. První se do proměnné `temp_distance` přiřadí vzdálenost od předchozího navigačního bodu, pokud neexistuje předchozí navigační bod, je `temp_distance` nastaven na hodnotu 200. Poté se do proměnné `distance`

přiřadí vzdálenost od aktuálního navigačního bodu. Dále je do proměnné `range` přiřazena vzdálenost, na kterou se přehrává zvuk navigace (Tabulka 3.1). Dále se ověří, zda je vzdálenost od aktuálního navigačního bodu menší než `range` a vzdálenost od předchozího navigačního bodu větší než `range` bez pěti metrů. Pokud jsou obě tyto podmínky splněny, přehraje se zvuk navigace a do proměnné `actualPoint` se přiřadí index následujícího navigačního bodu. Také se zde ověřuje, zda není konec trasy, pokud ano, tak se do proměnné `end` přiřadí hodnota `true` a navigace po trase skončí.

3.5 Výpočet navigačních bodů

Výpočet navigačních bodů probíhá vždy, když se naměřená trasa nebo informace o jízdě ukládají do databáze. Navigační body nelze v mobilní aplikaci upravovat, navigační body lze upravit pouze v informačním systému. Při výpočtu navigačních bodů se předpokládá, že jsou jednotlivé souřadnice trasy od sebe vzdáleny maximálně 50 metrů. Z GPS přijímače jsou souřadnice, v ideálním případě, načítány každých 5 metrů, ve skutečnosti se souřadnice načítají zhruba každých 10 - 30 metrů, podle aktuální rychlosti. Proto se může stát, pokud GPS přijímač často ztrácí signál, že bude výpočet navigačních bodů nepřesný.

Algoritmus pro výpočet navigačních bodů je obsáhlý, proto zde nebudu popisovat výpis zdrojového kódu. Výpis by musel být značně obsáhlý, aby v něm byly vidět všechny důležité operace, které se při výpočtu navigačních bodů provádějí, ale budu se zde snažit teoreticky vysvětlit, jak tento výpočet probíhá.

Trasa je rozdělena na sekce o maximální délce 500 metrů. Na začátku trasy je vytvořena první sekce, poté co je tato sekce ukončena, buď z důvodu maximální délky sekce, nebo z důvodu, že se změnil směr trasy, je vytvořena sekce nová. Tento princip vytváření nových sekcí probíhá dokud nejsou všechny souřadnice trasy přiřazeny do sekcí.

Každá sekce má přiřazen směr. Při vytvoření sekce, která neobsahuje žádné souřadnice, je tento směr nastaven na ROVNĚ. U každé sekce se také postupně vypočítává průměrný směr, jedná se o aritmetický průměr směrů souřadnic, přiřazených do této sekce. Poté jsou do sekce postupně přidávány souřadnice trasy. Při přidání nové souřadnice proběhne kotrola, zda nemá být sekce ukončena, sekce může být ukončena z důvodu maximální délky sekce, nebo z důvodu změny směru trasy. Tato kontrola je prováděna různými způsoby, podle toho jaký směr má sekce přiřazena.

Pokud je směr sekce nastaven na ROVNĚ, probíhá kontrola ukončení sekce následujícím způsobem. První se vypočítá směr nové souřadnice k následující souřadnici. Poté se vypočítá rozdíl mezi směrem nové souřadnice a průměrným směrem sekce. Pokud je

tento rozdíl mezi -10 a 10 stupni, je souřadnice přidána do sekce, a směr sekce zůstane nastaven na ROVNĚ. Pokud je tento rozdíl větší než 10 stupňů, naznačuje to, že se trasa začíná stáčet doleva. Aby se tento předpoklad potvrdil, proběhne ověření u následujících deseti souřadnic trasy, zda je rozdíl mezi jejich směry a průměrným směrem sekce také větší než 10 stupňů. Pokud minimálně jedna souřadnice tuto podmínku nesplňuje je nová souřadnice přidána do sekce a směr sekce zůstane nastaven na ROVNĚ. Pokud tuto podmínku splňuje všech deset souřadnic, opravdu se trasa začíná stáčet doleva. Pokud je počet souřadnic v sekci menší než 7, přidá se souřadnice do sekce a směr sekce je nastaven na VLEVO, pokud je počet souřadnic v sekci větší nebo roven 7, souřadnice se do sekce nepřidá a tato sekce je ukončena. Směr této sekce zůstane nastaven na ROVNĚ. Pokud je rozdíl mezi směrem nové souřadnice a průměrným směrem sekce menší než 10 stupňů, naznačuje to, že se směr trasy začíná stáčet doprava, aby se tento předpoklad potvrdil, proběhne ověření směrů následujících deseti souřadnic trasy, stejně jako je tomu v případě, že se trasa začíná stáčet doleva, s tím rozdílem, že se zjišťuje, zda je rozdíl jejich směrů a průměrného směru sekce menší než 10 stupňů.

Pokud je směr sekce nastaven na VLEVO, probíhá kontrola ukončení sekce následujícím způsobem. Také zde se první vypočítá směr nové souřadnice k následující souřadnici a je zde také vypočítán rozdíl mezi tímto směrem a průměrným směrem sekce, dále bude tento rozdíl nazýván jako hlavní rozdíl. Poté proběhne ověření směru následujících pěti souřadnic trasy, toto ověření probíhá následujícím způsobem. U každé souřadnice se vypočítá směr k následující souřadnici trasy, poté se vypočítá rozdíl mezi tímto směrem a průměrným směrem sekce. Pokud je rozdíl směrů všech pěti testovaných souřadnic menší než hlavní rozdíl, znamená to, že se trasa přestává stáčet doleva, proto je sekce ukončena a její směr zůstane nastaven na VLEVO. Pokud je rozdíl směrů alespoň u jedné testované souřadnice větší než hlavní rozdíl, předpokládá se, že se trasa stále stáčí doleva, aby se tento předpoklad potvrdil, proběhne zde ještě ověření, zda se následujících deset souřadnic trasy opravdu stáčí doleva. Pokud ano, přidá se souřadnice do sekce a její směr zůstane nezměněn. Pokud ne, sekce je ukončena a její směr zůstane nastaven na VLEVO. Toto ověření probíhá tak, že se spočítá průměrný směr těchto deseti souřadnic a pokud jsou všechny rozdíly mezi směry jednotlivých souřadnic a průměrným směrem těchto deseti souřadnic větší než 10 stupňů, je směr následujících deseti souřadnic VLEVO. Pokud jsou všechny tyto rozdíly menší než -10 stupňů, je směr následujících deseti souřadnic VPRAVO. A pokud není splněna ani jedna z těchto podmínek, je směr následujících deseti souřadnic ROVNĚ.

Pokud je směr sekce nastaven na VPRAVO, probíhá kontrola ukončení sekce stejným způsobem, jako když je směr sekce nastaven na VLEVO. Jediný rozdíl je v tom, že se

zjišťuje zda je rozdíl směrů všech pěti testovaných souřadnic větší než hlavní rozdíl.

První souřadnici v sekci se přiřadí navigační značka podle směru dané sekce. Další souřadnice v sekci jsou promazány, aby se do databáze neukládalo tak velké množství dat. Pokud je směr sekce ROVNĚ, ukládá se do databáze každá desátá souřadnice, včetně poslední souřadnice v sekci. Pokud je směr sekce nastaven na VLEVO nebo VPRAVO, ukládá se do databáze každá třetí souřadnice, včetně poslední souřadnice v sekci.

3.6 Volání WCF služeb

WCF služby jsou implementovány v informačním systému. Všechny operace, probíhající mezi mobilní aplikací a informačním systémem, jsou tvořeny z několika volání různých WCF služeb. Přesný popis jednotlivých operací najdete v kapitole Komunikace mobilní aplikace s informačním systémem.

Postup volání WCF služby z mobilní aplikace zde bude demonstrován na volání WCF služby `sendRoute(Route, int)` (Výpis 3.12), pomocí které se z mobilní aplikace posílají trasy na informační systém, tato WCF služba vrací identifikační značku trasy.

```
String SERVICE_URI = "http://cyklonavigace.aspone.cz/Service.svc";
HttpPost request = new HttpPost(SERVICE_URI + "/sendRoute?id=" + user_id);
request.setHeader("Accept", "application/json");
request.setHeader("Content-type", "application/json");
JSONStringer jsonRoute = new JSONStringer().object().key("route").object()
  .key("id").value(route.id)
  // přidání ostatních hodnot uložených ke trase
  .endObject().endObject();
request.setEntity(new StringEntity(jsonRoute.toString(), "utf-8"));
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpResponse response = httpClient.execute(request);
HttpEntity responseEntity = response.getEntity();
char[] buffer = new char[(int)responseEntity.getContentLength()];
InputStream stream = responseEntity.getContent();
InputStreamReader reader = new InputStreamReader(stream);
reader.read(buffer); stream.close();
String identifier = new String(buffer, 1, buffer.length - 2);
```

Výpis 3.12: Volání WCF služby, která odesílá trasu na informační systém

Nejprve se do proměnné `SERVICE_URI` přiřadí koncový bod, na kterém jsou WCF služby, poskytované informačním systémem, dostupné. Poté je vytvořena instance třídy `HttpPost`[9], jako parametr je zadána přesná adresa služby. Do hlavičky `HttpPost` nastaví typ posílaných dat, veškerá posílaná data pomocí WCF služeb jsou typu JSON.

Trasu nelze poslat jako instanci třídy `Route`, proto je vytvořena instance třídy `JSONStringer` pojmenovaná jako `jsonRoute`, do které se uloží hodnoty uložené ke trase. Je důležité, aby byly jednotlivé hodnoty uložené ke trase pojmenovány stejně jako v informačním systému. Poté se do `HttpPost` přidá jako entita i trasa uložená v proměnné `jsonRoute`. WCF služba je poté zavolána a výsledky, které služba vrací, jsou uloženy do proměné `response`, která je typu `HttpResponse`. Nakonec proběhne načtení výsledků uložených v proměnné `response`. Nejprve se výsledky převedou na `HttpEntity`, poté je vytvořen datový proud, který z `HttpEntity` načte jednotlivé bity, a nakonec jsou načtené bity převedeny na datový typ `String` a uloženy do proměnné `identifier`.

4 Popis informačního systému

V této části budou popsány všechny funkce informačního systému z uživatelského pohledu.

4.1 Zadání

Informační systém slouží pro vytváření a úpravu tras, a také pro sdílení tras mezi uživateli. U jednotlivých tras lze upravovat souřadnice, mazat, vytvářet, měnit typ a přesouvat navigační body, upravovat název a obtížnost trasy, a měnit typ kola, vhodný pro projetí trasy. Informační systém také zjišťuje ve kterých krajích se trasa nachází. Jednotlivé trasy lze zobrazit na mapě, u každé trasy lze navíc zobrazit graf výškového profilu. U informací o jízdě lze zobrazit graf rychlosti v závislosti na ujeté vzdálenosti a graf rychlosti v závislosti na čase jízdy. Trasu lze zveřejnit a po jejím zveřejnění si ji můžou jiní uživatelé přidat do svých tras. Veřejné trasy lze vyhledávat pomocí různých kritérií.

4.2 Popis funkcí informačního systému

Zde se zaměřím na popis funkcí informačního systému z uživatelského pohledu. Informační systém je spuštěn na adrese cyklonavigace.aspone.cz, pro správnou funkcionalitu všech níže popsaných funkcí je třeba mít nainstalovaný webový prohlížeč Google Chrome nebo Internet Explorer, ve webovém prohlížeči Mozilla Firefox nefunguje úprava trasy a úprava navigačních bodů.

Informační systém je rozdělen do tří částí (Veřejné trasy, Můj profil a Mapa). Do části Můj profil se dostane pouze přihlášený uživatel, tato část je dále rozdělena na tři menší celky (Moje trasy, Vložit trasu a Detail profilu). Níže budou popsány všechny funkce jednotlivých částí.

4.2.1 Veřejné trasy

V této části informačního systému budou zobrazeny všechny zveřejněné trasy, které jsou uloženy v databázi informačního systému. Zveřejněné trasy bude možné vyhledávat podle různých kritérií (Obrázek 4.1). Vyhledávací kritéria jsou následující: název trasy, obtížnost trasy, typ kola, vhodný na projetí trasy, délka trasy (rozmezí), kraj, ve kterém se trasa nachází, okolí bodu zvoleného na mapě, nastoupané metry (rozmezí) a naklesané metry (rozmezí).

U každé trasy, která splňuje vyhledávací kritéria, bude zobrazen její název, popis, obtížnost, typ kola, délka, průměrný sklon a nastoupané a naklesané metry. Dále bude

u každé trasy zobrazen malý obrázek mapy, na kterém bude vykreslena daná trasa. U každé trasy bude navíc zobrazeno tlačítko *Detail trasy*. Pokud bude uživatel přihlášen, tak se u tras, které nemá uživatel přidány ve svých trasách, navíc zobrazí tlačítko *Přidat do mých tras*.

Obrázek 4.1: Veřejné trasy

V detailu veřejné trasy bude ve vrchní části stránky zobrazena mapa, na které bude vykreslena trasa, na této mapě bude také možné zobrazit navigační body. Pod mapu bude tlačítko *Zpět*, kterým se uživatel může vrátit do výpisu veřejných tras, a pokud bude uživatel přihlášen a vybranou trasu nebude mít přidanou do svých tras, zobrazí se také tlačítko *Přidat trasu k mým trasám*. Dále budou následovat informace o trase, její název, popis, obtížnost, typ kola, výpis krajů, ve kterých se trasa nachází, délka, nastoupané a naklesané metry, průměrný sklon trasy a graf výškového profilu.

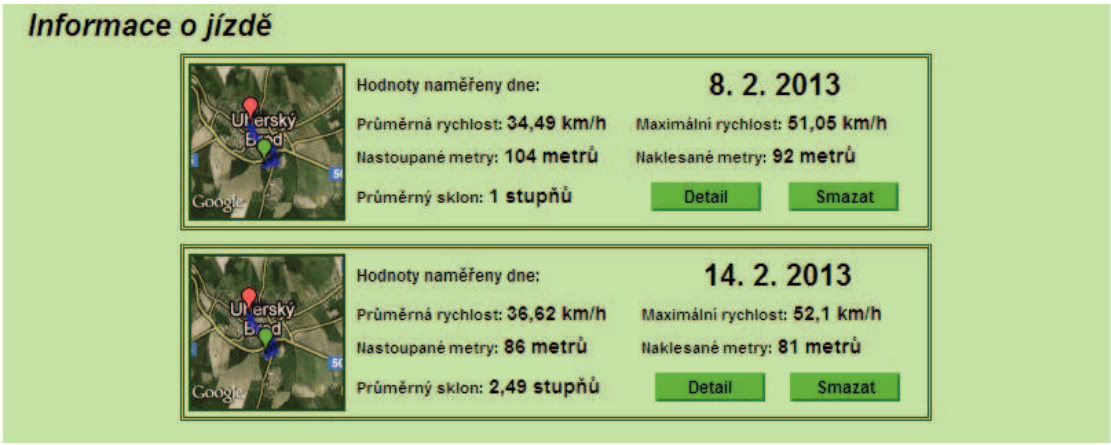
4.2.2 Moje trasy

Do této části informačního systému budou mít přístup pouze přihlášení uživatelé. Pokud se do této částí přesune nepřihlášený uživatel, bude vyzván k přihlášení se. Jsou zde

zobrazeny všechny trasy, které má uživatel přidány do svých tras.

U každé trasy bude zobrazen její název, obtížnost, popis, typ kola, délka, průměrný sklon, nastoupané a naklesané metry a také informace o synchronizaci trasy s mobilním zařízením. Trasa může být nahrána na mobilní zařízení, může čekat na nahrání na mobilní zařízení a nebo trasa není nahrána na mobilní zařízení. Dále bude u každé trasy zobrazen obrázek mapy, na kterém bude vykreslena konkrétní trasa. U každé trasy bude také tlačítko *Detail trasy* a tlačítko *Smazat trasu*.

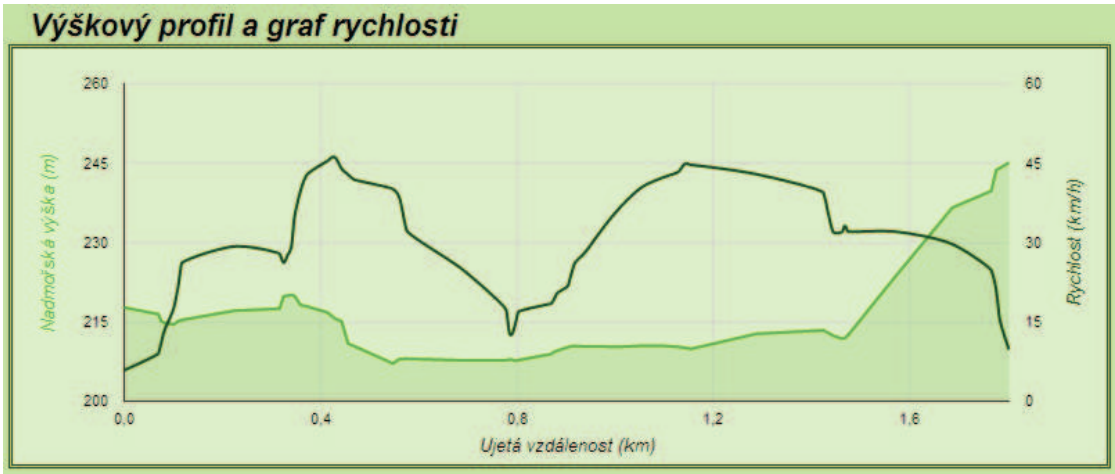
V detailu trasy bude ve vrchní části stránky zobrazena mapa, na které bude vykreslena trasa. Nad mapou bude tlačítko *Upravit navigační body* a tlačítko *Upravit souřadnice*. Po kliknutí na tlačítko *Upravit navigační body* se na mapě vykreslí navigační body, tyto body bude možné přesouvat a mazat, dále bude možné přidávat další navigační body. Po kliknutí na tlačítko *Upravit souřadnice* bude možné přesouvat a mazat jednotlivé souřadnice trasy, a také přidávat nové. Pod mapou bude zobrazeno tlačítko *Upravit trasu*, po kliknutí na toto tlačítko bude možné upravovat název trasy, popis trasy, obtížnost trasy a typ kola, dále zde bude tlačítko *Smazat trasu* a také tlačítko *Zpět*. Pod těmito tlačítky budou zobrazeny další informace o trase. Její název, popis, obtížnost, typ kola, kraje, ve kterých se trasa nachází, délka, nastoupané metry, naklesané metry, průměrný sklon a informace o synchronizaci trasy s mobilním zařízením, pokud není trasa nahrána na mobilní zařízení, také tlačítko *Nahrát trasu na mobilní zařízení*. Pokud není trasa zveřejněna, bude zde také zobrazeno tlačítko *Zveřejnit trasu*. Pod těmito informacemi o trase bude vykreslen graf výškového profilu a pod tímto grafem seznam informací o jízdě, uložených k této trase (Obrázek 4.2).



Obrázek 4.2: Seznam informací o jízdě

V detailu informací o jízdě bude ve vrchní části stránky zobrazena mapa, na které

budou vykresleny souřadnice naměřené při této jízdě. Na mapě bude možné zobrazit také navigační body. Pod mapou budou zobrazeny tlačítka *Smazat informace o jízdě* a *Zpět*. Pod těmito tlačítky budou zobrazeny základní informace o trase, ke které jsou informace o jízdě naměřeny (název, popis, obtížnost a typ kola), a dále informace o jízdě (datum, kdy byly informace naměřeny, ujetá vzdálenost, čas jízdy, průměrná rychlost, maximální rychlost, nastoupané metry, naklesané metry a průměrný sklon). Poté bude následovat graf výškového profilu kombinovaný s grafem rychlosti v závislosti na ujeté vzdálenosti (Obrázek 4.3) a graf rychlosti v závislosti na čase jízdy.



Obrázek 4.3: Graf výškového profilu s grafem rychlosti

4.2.3 Vložit trasu

Do této části informačního systému mají přístup pouze přihlášení uživatelé. Pokud se do této části přesune nepřihlášený uživatel, bude vyzván k přihlášení se. Tato část slouží pro vkládání nových tras. Novou trasu může vložit pouze přihlášený uživatel, nové trasy jsou ukládány do tras uživatele. Nová trasa může být poté zveřejněna nebo také nahrána na mobilní zařízení.

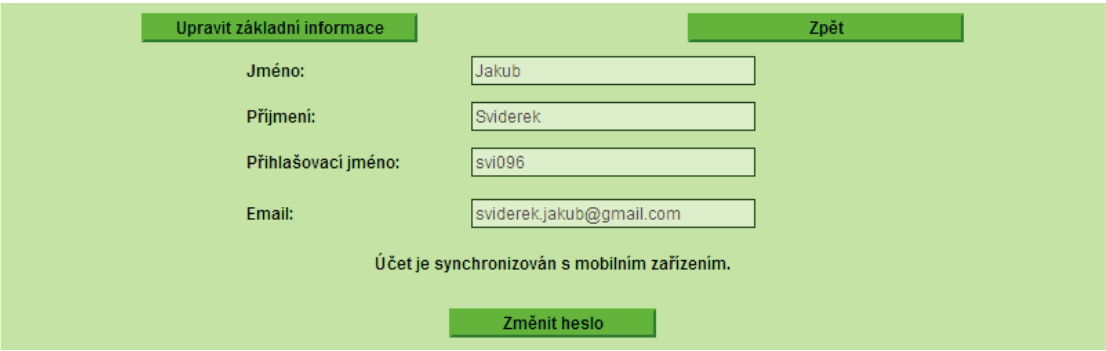
Trasu bude možné vytvořit třemi způsoby. První způsob vytvoření trasy je vyhledání trasy na mapě. Uživatel zde zadá pouze počáteční a koncový bod trasy, a trasa se mezi těmito body automaticky vyhledá, do trasy bude možné přidat až osm kontrolních bodů. Další způsob vytvoření trasy bude nahrání souřadnic ze souboru, souřadnice bude možné načíst ze souborů typu KML. Načtené souřadnice bude možné dále upravovat. Poslední způsob vytvoření trasy bude ruční zadání souřadnic, uživatel postupně na mapě nakliká všechny souřadnice trasy.

Po přesunu do této části se ve vrchní části obrazovky zobrazí tlačítka, kde každé tlačítko bude reprezentovat jeden způsob vytvoření nové trasy, tento panel tlačítek se bude měnit podle toho, ve kterém režimu vytváření trasy se bude uživatel nacházet. Pokud uživatel potvrdí zadané souřadnice, zobrazí se mu tlačítko *Upravit navigační body*, po kliknutí na toto tlačítko bude moci upravovat navigační body stejným způsobem, jako v detailu trasy, a také tlačítka *Upravit souřadnice* a *Smazat souřadnice z mapy*. Pod tímto panelem tlačítek bude zobrazena mapa, na které se budou vykreslovat zadané souřadnice.

Pod mapou bude zobrazeno tlačítko *Uložit trasu*, trasu nebude možné uložit dokud nebudou zadány souřadnice trasy a dokud nebude vyplněn její název, obtížnost a typ kola, a také tlačítko *Zpět*. Pod těmito tlačítky bude zobrazen formulář pro zadání názvu trasy, popisu trasy, obtížnosti trasy a typu kola. Pod formulářem bude vypsána délka trasy, nastoupané a naklesané metry a průměrný sklon trasy. Ve spodní části stránky bude tlačítko *Zobrazit výškový profil*, po kliknutí na toto tlačítko se zobrazí graf výškového profilu, tímto tlačítkem bude možné tento graf opět skrýt.

4.2.4 Detail profilu

Do této části informačního systému mají přístup pouze přihlášení uživatelé. Pokud se do této části přesune nepřihlášený uživatel, bude vyzván k přihlášení se. Tato část slouží pro změnu informací o uživateli, včetně přihlašovacích údajů (Obrázek 4.4).



Obrázek 4.4: Detail profilu

Ve vrchní části stránky bude tlačítko *Upravit základní informace*, po kliknutí na toto tlačítko bude možné změnit jméno, příjmení, přihlašovací jméno a email, před uložením do databáze bude zkontrolováno, zda zadané přihlašovací jméno a email nepoužívá jiný uživatel, a také tlačítko *Zpět*. Pod těmito tlačítky bude zobrazeno jméno, příjmení, přihlašovací jméno a email uživatele. Dále zde bude zobrazena informace o synchronizaci

uživatelského účtu s mobilním zařízením. Ve spodní části stránky bude zobrazeno tlačítko *Změnit heslo*, po kliknutí na toto tlačítko se zobrazí formulář pro změnu hesla. Nové heslo musí obsahovat minimálně 8 znaků, pokud uživatel zadá špatné staré heslo, může si vygenerovat nové heslo, které mu bude zasláno na email. Hesla se do databáze budou ukládat zašifrovaná.

4.2.5 Mapa

Tato část informačního systému slouží pro vytváření nových tras, nové trasy zde budou moci vytvářet i nepřihlášení uživatelé, avšak pro uložení trasy se budou muset přihlásit, aby neregistrovaní uživatelé nevytvářeli nesmyslné trasy.

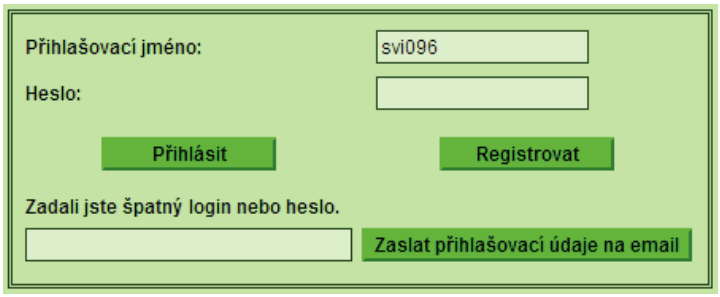
Ve vrchní části obrazovky bude zobrazen panel tlačítek a mapa, stejně jako v části Vložit trasu. Pod mapou bude zobrazena délka trasy, nastoupané metry, naklesané metry a průměrný sklon trasy. Ve spodní části obrazovky bude tlačítko *Uložit trasu*, které uživatele přesměruje do sekce Vložit trasu, kde bude moci navolenou trasu uložit mezi své trasy. A pokud bude na mapě zobrazena nějaká trasa, tak i tlačítko *Zobrazit výškový profil trasy*, po kliknutí na toto tlačítko se zobrazí graf výškového profilu, tímto tlačítkem bude možné tento graf opět skrýt.

4.2.6 Registrace nového uživatele a přihlášení

Registrace nového uživatele bude možná po kliknutí na tlačítko *Registrovat* v přihlašovacím formuláři. Poté se zobrazí formulář pro registraci nového uživatele. Uživatel bude vyzván k vyplnění jména, příjmení, přihlašovacího jména, pod kterým se bude přihlašovat, email, na který bude zaslán aktivační odkaz a na který si bude moci uživatel poslat nové heslo, v případě, že ho zapomněl, a také k vyplnění hesla, heslo musí obsahovat minimálně 8 znaků. Před registrací se bude kontrolovat, zda zadané přihlašovací jméno a email nepoužívá jiný uživatel, a také, zda je email zadán ve správném formátu. Po úspěšné registraci bude na zadanou emailovou adresu zaslán email, ve kterém budou uživateli zaslány přihlašovací údaje a také aktivační odkaz. Dokud nebude účet aktivován, uživatel se nebude moci přihlásit.

Přihlašovací formulář se bude zobrazovat po kliknutí na tlačítko *Přihlásit*, které bude umístěno ve vrchní části všech stránek, toto tlačítko bude nahrazeno tlačítkem *Odhlásit*, pokud se uživatel přihlásí. Nalevo od tohoto tlačítka bude zobrazeno jméno a příjmení přihlášeného uživatele, pokud uživatel nebude přihlášen bude zde zobrazen text „nepřihlášen“. Přihlašovací formulář se uživateli zobrazí také v případě, že nebude přihlášen a pokusí se dostat do sekce Můj profil.

Pokud uživatel zadá špatné přihlašovací údaje, bude mu, po vyplnění správného emailu, vygenerováno nové heslo a zasláno na jeho email (Obrázek 4.5). Po úspěšném přihlášení bude uživatel přesměrován na poslední zobrazenou stránku.



Obrázek 4.5: Vygenerování nového hesla v přihlašovacím formuláři

5 Implementace informačního systému

V této části budou popsány hlavní funkce informačního systému z programátorského pohledu. Budou zde zjednodušeně popsány důležité funkce, jako je práce s databází, implementace WCF služeb, algoritmus pro výpočet navigačních bodů, práce s mapou, Google Elevation API, Google Static Map API a algoritmus pro určení krajů, ve kterých se trasa nachází.

5.1 Struktura informačního systému

Informační systém je rozdělen na tři části (`Cyklonavigace-IS`, `DBConnect`, `Services`). V části `Cyklonavigace-IS` jsou implementovány všechny funkce jednotlivých webových stránek. Část `DBConnect` je knihovna, která slouží pro přístup k datům v databázi. Část `Services` je knihovna, ve které jsou naimplementovány všechny WCF služby. Dále bude popsán obsah jednotlivých částí informačního systému.

- **`Cyklonavigace-IS`** - V této části jsou implementovány všechny funkce jednotlivých webových stránek.
  - **`App_Code`** - V tomto adresáři jsou uloženy pomocné třídy sloužící pro výpočet navigačních bodů, zjišťování nadmořské výšky jednotlivých souřadnic a načtení souřadnic ze souboru.
  - **`App_Data`** - V tomto adresáři je uložena databáze, kterou informační systém používá pro ukládání dat. Databáze je uložena v souboru typu mdf.
  - **`App_Themes`** - V tomto adresáři jsou uloženy soubory definující vzhled jednotlivých internetových stránek informačního systému.
  - **`Bin`** - V tomto adresáři jsou uloženy knihovny použité v informačním systému.
  - **`Img`** - V tomto adresáři jsou uloženy obrázky použité v informačním systému, včetně obrázků obtížnosti trasy a markerů zobrazených na mapě.
  - **`user`** - V tomto adresáři jsou uloženy internetové stránky, do kterých má přístup pouze přihlášený uživatel.
- **`DBConnect`** - Tato knihovna slouží pro manipulaci s daty uloženými v databázi informačního systému.
  - **`DBO`** - Tento adresář obsahuje, pro každou tabulku v databázi, jednu třídu, pomocí které se manipuluje s daty uloženými v databázi.

  - **`Models`** - Tento adresář obsahuje, pro každou tabulku v databázi, jednu třídu, která reprezentuje datovou strukturu dané tabulky.

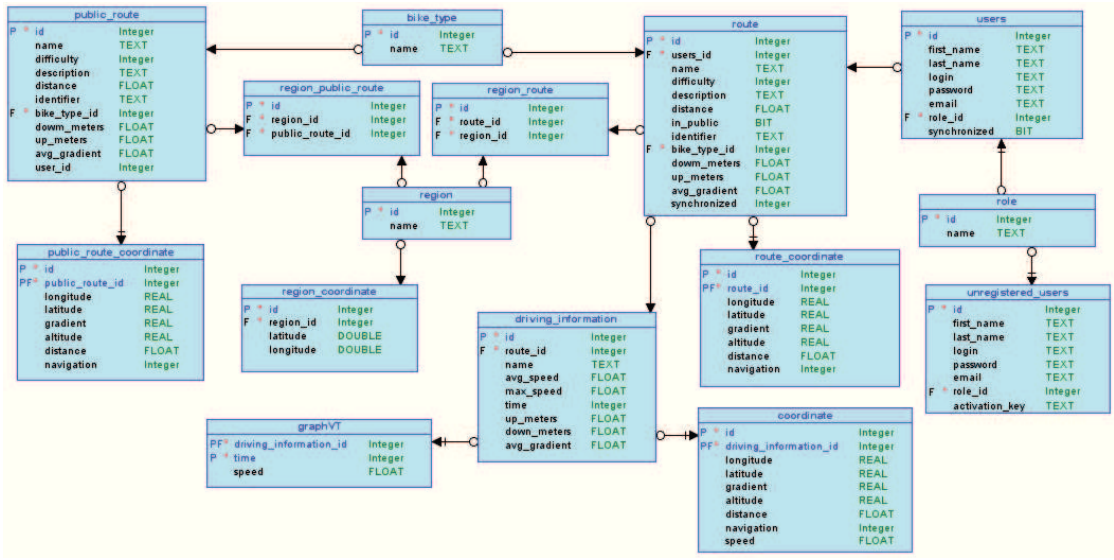
- **`Services`** - V této knihovně jsou implemetonvány všechny WCF služby, které jsou poskytovány informačním systémem.

5.2 Databáze

Informační systém používá pro ukládání dat databázi vytvořenou v programu MS SQL Server 2008[10]. Pro přístup do databáze je naimplementována knihovna `DBConnect`.

5.2.1 Struktura databáze

Databáze obsahuje celkem patnáct tabulek (Obrázek 5.1). Dvanáct z nich slouží pro ukládání tras a informací o jízdě, a tři tabulky slouží pro ukládání uživatelů,



Obrázek 5.1: Schéma databáze informačního systému

- **public_route** - Tabulka, do které se ukládají veřejné trasy. Ke každé trase se ukládá ID, název, obtížnost, popis, délka, identifikační značka trasy, kterou má každá trasa jedinečnou a podle které probíhá synchronizace tras s mobilním zařízením, typ kola vhodného pro projetí trasy, naklesané metry, nastoupané metry, průměrný sklon trasy a ID uživatele, který trasu zveřejnil.

- **public_route_coordinate** - Tabulka, do které se ukládají souřadnice k jednotlivým veřejným trasám. Ke každé souřadnici se ukládá ID trasy, ke které je souřadnice

naměřena, ID souřadnice, které určuje pořadí souřadnic v jakém byly naměřeny, u každé trasy jsou číslovány od 1, zeměpisná délka (longitude), zeměpisná šířka (latitude), aktuální sklon, nadmořská výška, vzdálenost od první souřadnice trasy a informace, zda je souřadnice navigačním bodem.

- **route** - Tabulka, do které se ukládají trasy uživatelů. Ke každé trase se ukládá ID, ID uživatele, který danou trasu vlastní, název, obtížnost, popis, délka, informace, zda je trasa zveřejněna, identifikační značka trasy, kterou má každá trasa jedinečnou a podle které probíhá synchronizace tras s mobilním zařízením, ID typu kola vhodného pro projetí trasy, naklesané metry, nastoupané metry, průměrný sklon a informace, zda je trasa synchronizována s mobilním zařízením.

- **route_coordinate** - Tabulka, do které se ukládají souřadnice k jednotlivým trasám uživatelů. Ke každé souřadnici se ukládá ID trasy, ke které je souřadnice naměřena, ID souřadnice, které určuje pořadí souřadnic v jakém byly naměřeny, u každé trasy jsou číslovány od 1, zeměpisná délka (longitude), zeměpisná šířka (latitude), aktuální sklon, nadmořská výška, vzdálenost od první souřadnice trasy a informace, zda je souřadnice navigačním bodem.

- **driving_information** - Tabulka, do které se ukládají hodnoty k informacím o jízdě. Ukládá se zde ID, název, což je datum, kdy byly hodnoty naměřeny, ID trasy, ke které jsou informace o jízdě uloženy, průměrná rychlost, maximální rychlost, čas jízdy, nastoupané metry, naklesané metry a průměrný sklon.

- **coordinate** - Tabulka, do které se ukládají souřadnice naměřené k jednotlivým informacím o jízdě. Ke každé souřadnici se ukládá ID informací o jízdě, ke kterým je souřadnice naměřena, ID souřadnice, které určuje pořadí souřadnic v jakém byly naměřeny, u každé trasy jsou číslovány od 1, zeměpisná délka (longitude), zeměpisná šířka (latitude), aktuální sklon, nadmořská výška, vzdálenost od první souřadnice trasy, informace zda je souřadnice navigačním bodem a rychlost.

- **graphVT** - Tabulka, do které se ukládají body, potřebné pro vykreslení grafu rychlosti v závislosti na čase jízdy. Ke každému bodu se ukládá ID informací o jízdě, ke kterým je bod uložen, čas a rychlost.

- **bike_type** - Tabulka, která reprezentuje číselník typů kol. V této tabulce je uložen název a ID každého typu kola.

- **region** - Tabulka, která reprezentuje číselník krajů České republiky. V této tabulce je uložen název a ID každého kraje.

- **region_coordinate** - Tabulka, ve které jsou uloženy souřadnice hranic krajů. Ke každé souřadnici je uloženo ID kraje, zeměpisná šířka (latitude) a zeměpisná délka (longitude).
- **region_public_route** - Tabulka, do které se ukládají kraje, ve kterých leží jednotlivé veřejné trasy. Do tabulky jsou ukládány dvojice ID veřejné trasy a ID kraje.
- **region_route** - Tabulka, do které se ukládají kraje, ve kterých leží jednotlivé trasy uživatelů. Do tabulky jsou ukládány dvojice ID trasy a ID kraje.
- **users** - Tabulka, do které se ukládají informace o registrovaných uživatelích. Ke každému uživateli je uloženo jeho ID, jméno, příjmení, přihlašovací jméno, heslo hashované pomocí algoritmu SHA1, email, ID role uživatele a informace o synchronizaci s mobilním zařízením.
- **unregistred_users** - Tabulka, do které se ukládají informace o uživatelích, kteří se registrovali, ale zatím neprovedli aktivaci uživatelského účtu. Ke každému uživateli se ukládá ID, jméno, příjmení, přihlašovací jméno, heslo, hashované pomocí algoritmu SHA1, email, ID role a aktivační klíč.
- **role** - Tabulka, která reprezentuje číselník rolí uživatelů. V této tabulce je uložen název a ID všech rolí, které můžou mít uživatelé přiřazeny.

5.2.2 Přístup k datům v databázi

Pro každou tabulku jsou v knihovně `DBConnect` definovány dvě třídy, jedna třída, z balíčku `Models`, reprezentuje strukturu dané tabulky, a ve druhé třídě, z balíčku `DBO`, jsou definovány všechny operace prováděné nad konkrétní tabulkou.

Pro manipulaci s daty uloženými v databázi je použita třída `SqlConnection`[11] (Výpis 5.1). Nejprve je vytvořena instance této třídy, jako parametr je zadán připojovací řetězec, který je uložen v konfiguračním souboru informačního systému `web.config`[12]. Poté je vytvořena instance třídy `SqlCommand`, pomocí které se nad databází provádí Sql příkazy. Dále se vytvoří Sql příkaz a otevře se databáze. Nakonec je proveden Sql příkaz a výsledky tohoto příkazu jsou předány `SqlDataReaderu`, poté se tyto výsledky převedou na datový typ `PublicRoute`. Pokud je počet načtených veřejných tras roven 0, metoda vrátí false, v opačném případě vrací metoda list veřejných tras.

```
List<PublicRoute> results = new List<PublicRoute>();
using (SqlConnection cnt=new SqlConnection(ConfigurationManager.ConnectionStrings["localCS"].
     ConnectionString)) {
  SqlCommand cmd = cnt.CreateCommand();
  cmd.CommandText = //vytvoření SQL dotazu
  cnt.Open();
  SqlDataReader reader = cmd.ExecuteReader();
  while (reader.Read()) {
    results.Add(new PublicRoute() {
      //načtení hodnot z SqlDataReaderu
    });
}}
if (results.Count > 0)
  return results;
else
  return null;
```

Výpis 5.1: Ukázka načtení veřejných tras z databáze

5.3 Implementace WCF služeb

Pro komunikaci mezi mobilní aplikací a informačním systémem jsou použity WCF služby [13]. Tyto služby jsou implementovány v knihovně `Services`. Tato knihovna obsahuje dvě třídy (`IService.cs` a `Service.cs`). Ve třídě `IService.cs` jsou definována rozhraní jednotlivých WCF služeb (Výpis 5.2). Každé rozhraní WCF služby musí být označeno jako `[OperationContract]`, a dále se u každého rozhraní nastavují vlastnosti dané WCF služby. Mezi tyto vlastnosti patří způsob posílání dat (`Method`), v ukázce je zvolena metoda POST, HTML adresa, na které bude daná WCF služba dostupná (`UriTemplate`), typ posílané zprávy (`BodyStyle`), formát přijatých a odeslaných dat (`ResponseFormat` a `RequestFormat`), všechna data, posílána pomocí WCF služeb, jsou typu JSON.

```
[OperationContract]
[WebInvoke(Method = "POST",
  UriTemplate = "/sendRoute?id={user_id}",
  BodyStyle = WebMessageBodyStyle.WrappedRequest,
  ResponseFormat = WebMessageFormat.Json,
  RequestFormat = WebMessageFormat.Json)]
string sendRoute(Route route, int user_id);
```

Výpis 5.2: Ukázka rozhraní WCF služby

Ve třídě `Service.cs` je definována funkčnost jednotlivých WCF služeb. WCF služby používají pro manipulaci s databází informačního systému knihovnu `DBConnect`.

Pomocí WCF služeb můžou být posílány i vlastní datové typy. Vlastní datové typy, které chceme posílat pomocí WCF služeb musí být označeny jako `[DataContract]`, a jednotlivé členské proměnné musí být označeny jako `[Datamember]` a musí k nim být přiřazen název (Výpis 5.3). V ukázce lze vidět definici datového typu `Route`, a definici členských proměnných `id` a `name`, tímto způsobem jsou definovány všechny členské proměnné.

```
[DataContract]
public class Route {
  [DataMember(Name = "id")]
  public int id { get; set; }
  [DataMember(Name = "name")]
  public string name { get; set; }
}
```

Výpis 5.3: Definice vlastního datového typu posílaného pomocí WCF služby

V poslední řadě je nutné nastavit koncový bod, ze kterého budou WCF služby dostupné (Výpis 5.4). Vlastnosti koncového bodu se nastavují v konfiguračním souboru informačního systému `web.config`. Koncový bod, neboli EndPoint, se skládá ze tří částí (adresa, binding a kontrakt). Adresa specifikuje kam jsou zprávy posílané. Binding specifikuje komunikační mechanismus a popisuje způsob, jakým mají být zprávy posílané, binding specifikuje způsob přenosu, kódování, zabezpečení a podporu transakcí, v informačním systému je použit `webHttpBinding`. Kontrakt specifikuje rozhraní WCF služeb, zadává se ve formátu knihovna.název třídy.

```
<system.serviceModel>
  <services>
    <service name="Services.Service">
      <endpoint
        address="cyklonavigace.aspone.cz/Service.svc"
        binding="webHttpBinding"
        contract="Services.IService"
        behaviorConfiguration="httpBehavior" />
    </service>
  </services>
</system.serviceModel>
```

Výpis 5.4: Nastavení koncového bodu WCF služeb

5.4 Práce s mapou

V infomačním systému jsou použity mapové podklady, poskytované společností Google[14]. Pro práci s Google mapami je přednostně určen programovací jazyk JavaScript. Já jsem použil uživatelský ovládací prvek `Artem.Google`[15], jedná se o volně dostupnou knihovnu, pomocí které lze pracovat s Google mapou bez použití Javascriptu. V této knihovně jsou definovány třídy pro všechny objekty, které lze na mapu vykreslit, samotná mapa je definována jako uživatelský ovládací prvek. S mapou se pracuje pohodlně pomocí C# příkazů, ale našel jsem pár nedostatků, které mi trošku komplikovaly implementaci informačního systému. Hlavním nedostatkem této knihovny je, že jsou špatně zpracovávány události vyvolané manipulací s mapou a objekty, které jsou na ní vykresleny, v internetovém prohlížeči Mozilla Firefox. Bohužel jsem na to přišel pozdě, informační systém jsem testoval v internetových prohlížečích Google Chrome a Internet Explorer, proto je práce s mapou v informačním systému optimalizována pouze na dva výše zmíněné internetové prohlížeče.

Mapa se na stránku vkládá stejným způsobem jako ostatní ovládací prvky (Výpis 5.5). Nastavuje se zde ID ovládacího prvku (`ID`), klíč (`Key`), který lze bezplatně získat na internetových stránkách *http://www.google.com/apis/maps/signup.html*, výška (`Width`) a šířka (`Height`) mapy, pro jeho získání je nutné mít účet na Googlu, zeměpisná šířka (`Latitude`) a zeměpisná délka (`Longitude`) středu mapy, a přiblížení mapy (`Zoom`).

```
<artem:GoogleMap
  ID = "googlemap" runat = "server"
  Key= //Google map API klíč
  Width = "790px" Height = "600px"
  Latitude = "49.823809" Longitude = "15.53302"
  Zoom = "7">
</artem:GoogleMap>
```

Výpis 5.5: Vložení mapy na stránku

5.4.1 Vykreslení trasy a značek na mapě

Trasa lze na mapu vykreslit pomocí třídy GooglePolyline, značky se na mapu vykreslují pomocí třídy GoogleMarkers. Pro vykreslení trasy či značek je nutné obě tyto třídy nejdříve vložit na stránku (Výpis 5.6). U každé z nich se nastavuje ID ovládacího prvku (`ID`) a ID mapy, na kterou se mají vykreslit (`TargetControlID`).

```
<artem:GooglePolyline
  ID = "googlepolyline" runat = "server"
  TargetControlID = "googlemap">
</artem:GooglePolyline>
<artem:GoogleMarkers
  ID = "googlemarkers" runat = "server"
  TargetControlID = "googlemap">
</artem:GoogleMarkers>
```

Výpis 5.6: Definice GooglePolyline a GoogleMarkers

Při vykreslování trasy na mapu jsem se setkal s jedním problémem. Při každém opětovném načtení stránky se umaže poslední souřadnice trasy. Nezjištil jsem důvod, proč se poslední souřadnice umazává, proto je poslední souřadnice v poli souřadnic vložena dvakrát, a při každém opětovném načtení se v metodě `Page_Init` zduplikuje poslední souřadnice trasy a přidá se na konec pole souřadnic.

Samotné vykreslení trasy a značek na mapě probíhá následujícím způsobem (Výpis 5.7). Souřadnice trasy jsou přidávány do `googlepolyline` a značky jsou přidávány do pole značek uložených v `googlemarkers`. Se samotnou mapou není nutné manipulovat, protože obě tyto třídy mají definovánu mapu, na které se mají jednotlivé prvky vykreslit, v proměnné `TargetControlID`.

```
googlepolyline.Path.Clear();
googlemarkers.Markers.Clear();
for (int i = 0; i < coords.Count - 1; i++) {
  if (i == 0) {
    MyMarker marker = new MyMarker(new LatLng(coords[i].latitude, coords[i].longitude));
    //nastavení vlastností značky
    googlemarkers.Markers.Add(marker);
  } else if (i == coords.Count - 2) {
    MyMarker marker = new MyMarker(new LatLng(coords[i].latitude, coords[i].longitude));
    //nastavení vlastností značky
    googlemarkers.Markers.Add(marker);
  }
  LatLng point = new LatLng(coords[i].latitude, coords[i].longitude);
  googlepolyline.Path.Add(point);
}
```

Výpis 5.7: Vykreslení trasy a značek na mapě

Nejprve se vymažou body trasy uložené v `googlepolyline` a také pole značek v `googlemarkers`. Poté je definován cyklus, který prochází všechny souřadnice trasy, kromě poslední, ta je totiž stejná jako předposlední souřadnice, uložené v poli `coords`.

Pokud se jedná o první nebo předposlední souřadnici trasy, je vytvořena značka. Při vytváření značky se vytvoří instance třídy `LatLng`, která reprezentuje souřadnice, na kterých má být značka zobrazena. Poté je tato značka přidána do pole `googlemarkers`. Pro každou souřadnici trasy, uloženou v poli `coords`, je vytvořena instance třídy `LatLng`, a poté je tento bod přidán do bodů trasy uložených v `googlepolyline`.

5.5 Google Elevation API

Pro zjišťování nadmořské výšky souřadnic je v informačním systému použito Google Elevation API[16]. Nadmořská výška se zjišťuje u souřadnic tras, které jsou do databáze vloženy pomocí informačního systému, a také u souřadnic tras, které jsou nahrány z mobilní aplikace. V mobilní aplikaci se z GPS přijímače načítá nadmořská výška zaokrouhlená na metry, což způsobuje zkreslený výpočet sklonu trasy, proto je při nahrání trasy na informační systém nadmořská výška u každé souřadnice přepočítána.

Zjišťování nadmořské výšky probíhá následujícím způsobem (Výpis 5.8). Pomocí jednoho dotazu může být zjištěna nadmořská výška maximálně dvaceti souřadnic. Nejprve se poskládá dotaz, do dotazu se vždy vloží maximálně dvacet souřadnic. Poté se pomocí instance třídy `WebClient` získají výsledky dotazu. Vrácené výsledky jsou ve formátu Json, tyto výsledky jsou poté pomocí knihovny `Newtonsoft.Json` rozparsovány, a nadmořské výšky přiřazeny jednotlivým souřadnicím. Po každém odeslání dotazu a rozparsování výsledků je metoda na 150 milisekund uspána, Google Elevation API je trošku pomalejší a stávalo se, že některé dotazy nebyly úspěšně vyřízeny.

```
List<RouteCoordinate> results = new List<RouteCoordinate>();
for (int i = 0; i < coordinates.Count; i += 20) {
  string uri = "http :// maps.googleapis.com/maps/api/elevation/json?locations=";
  // přidání jednotlivých souřadnic do uri
  var page = new WebClient().DownloadString(uri);
  JObject o = (JObject)JsonConvert.DeserializeObject(page);
  int j = 0;
  foreach (var result in o["results"].Children()) {
    coordinates[i + j].id = i + j;
    coordinates[i + j].altitude = double.Parse(result.Value<string>("elevation").ToString(),
    System.Globalization.NumberStyles.Any, CultureInfo.InvariantCulture);
    results.Add(coordinates[i + j]);
    j++;
  }
  Thread.Sleep(150);
}
```

Výpis 5.8: Zjištění nadmořské výšky souřadnic trasy

5.6 Google Maps Image API

V seznamu veřejných i uživatelských tras, a také v seznamu informací o jízdě, je v informačním systému zobrazen obrázek mapy, na kterém je vykreslena tato trasa. Tyto obrázky jsou generovány pomocí Google Maps Image API[17].

Generování obrázku mapy probíhá následujícím způsobem (Výpis 5.9). Nejprve je nastaven střed mapy, jako střed mapy je zvolena souřadnice, která se nachází přesně v půli trasy, a také se podle délky trasy nastaví přiblížení mapy. Poté se na mapu přidají značky začátku a konce trasy, značky jsou umístěny na první a poslední souřadnici trasy. Nakonec jsou určeny souřadnice, které se na mapě mají zobrazit. Google Maps Image API v bezplatné verzi umožňuje na mapě zobrazit pouze 50 souřadnic. Pokud je počet souřadnic trasy menší než 50, zobrazí se na mapu všechny souřadnice trasy. Pokud je počet souřadnic trasy větší než 50, vypočítá se do proměnné `temp` hodnota, která se v každém kroku přičítá k indexu zobrazované souřadnice. Tím je zajištěno, že u libovolného počtu souřadnic trasy, bude vždy na mapě zobrazeno pouze 50 souřadnic rovnoměrně rozprostřených po trase.

```
string uri = "http :// maps.googleapis.com/maps/api/staticmap?sensor=false&center=";
float dist = coords[coords.Count − 1].distance / 2;
//přidání souřadnice středu mapy do uri, jedná se o souřadnici v půlce trasy
//nastavení přiblížení a velikosti obrázku, přiblížení se nastavuje podle délky trasy
//přidání značky na začátek a konec trasy
double temp = coordinates.Count / 50;
if (temp <= 1) {
  //přidání všech souřadnic do uri
} else {
    double i_temp = 0;
    while (i_temp < coordinates.Count − 1) {
      int i = (int)Math.Round(i_temp, 0);
      //přidání souřadnice na indexu i do uri
      i_temp += temp;
}}
image_map.ImageUrl = uri;
```

Výpis 5.9: Ukázka použití Google Static Map API

5.7 Google Chart API

Informační systém, stejně jako mobilní aplikace, zobrazuje u jednotlivých tras graf výškového profilu, a u informací o jízdě, zobrazuje graf výškového profilu spolu s grafem rychlosti v závislosti na ujeté vzdálenosti a graf rychlosti v závislosti na čase jízdy. Všechny

tyto grafy jsou vykresleny pomocí Google Chart API[18]. Vykreslování grafů pomocí Google Chart API probíhá pomocí programovacího jazyka JavaScript.

Pro zobrazení libovolného grafu pomocí Google Chart API je nutné v hlavičce stránky, na které má být tento graf zobrazen, definovat skript, který má v atributu `src` nastaven hodnotu *https://www.google.com/jsapi*, a také je nutné, aby byl na stránku vložen ovládací prvek `ScriptManager`, který má v parametru `EnablePageMethods` nastavenu hodnotu true. Dále je ve zdrojovém kódu definována metoda `getCoords()`, která vrací pole obsahující body potřebné pro vykreslení grafu. Tato metoda musí být označena jako `[System.Web.Services.WebMethod, ScriptMethod]`, aby jí bylo možné volat z aspx souboru, ve kterém je definován vzhled stránky.

Samotné vykreslení grafu probíhá způsobem znázorněným v ukázce (Výpis 5.10), jedná se o vykreslení grafu výškového profilu trasy.

```
google.load(" visualization ", "1", { packages: ["corechart"] });
google.setOnLoadCallback(getCoordinates);
function getCoordinates() {
  PageMethods.getCoords(drawChart);
}
function drawChart(result) {
  var temp = new Array();
  for (var i = 0; i < result.length; i++) {
    var description = //nastavení popisu bodu, který se zobrazí po najetí na tento bod
    var row = [result[i].distance / 1000, result[i]. altitude , description];
    temp[i] = row;
  }
  var data = new google.visualization.DataTable();
  data.addColumn('number', 'Vzdalenost');
  data.addColumn('number', 'Nadmorska vyska');
  data.addColumn({ type: 'string', role: ' tooltip ' });
  data.addRows(temp);
  var options = //nastavení vlastností grafu
  var chart = new google.visualization.AreaChart(document.getElementById('chart_div'));
  chart.draw(data, options);
}
```

Výpis 5.10: Vykreslení grafu pomocí Google Chart API

Pomocí prvního řádku jsou načteny balíčky potřebné pro vykreslení grafu. Druhý řádek zavolá, po načtení stránky, funkci `getCoordinates()`, v těle této funkce je zavolána metoda `getCoords()`, která vrací pole, obsahující body potřebné pro vykreslení grafu, které je předáno jako parametr `result` funkci `drawChart()`. Ve funkci

`drawChart()` je nejprve vytvořeno pomocné pole `temp`, do kterého se ukládají jednotlivé body grafu. Ke každému bodu je uložena vzdálenost od první souřadnice trasy, nadmořská výška a popis, který se zobrazí po najetí na tento bod. V popisu je zobrazena vzdálenost od první souřadnice trasy, nadmořská výška a také aktuální sklon trasy. Poté je vytvořena datová tabulka `data`, ve které je definován popis a typ jednotlivých sloupců a poté jsou do této tabulky vloženy body, které jsou uložené v poli `temp`. Dále jsou v proměnné `options` definovány vlastnosti grafu, a do proměnné `chart` je přiřazeno ID oddílu, na který má být graf vykreslen. Pomocí posledního řádku se vykreslí graf.

5.8 Výpočet navigačních bodů

Výpočet navigačních bodů tras, které jsou vytvořeny pomocí informačního systému, probíhá jiným způsobem, jako výpočet navigačních bodů tras, které jsou naměřeny pomocí mobilní aplikace. Mobilní aplikace načítá souřadnice trasy, v ideálním případě, každých 5 metrů. Ve skutečnosti jsou od sebe souřadnice vzdáleny 10 až 30 metrů, podle aktuální rychlosti. Souřadnice, které jsou načteny z `GoogleDirection`, pomocí kterého probíhá vyhledání trasy, jsou od sebe v rovných úsecích trasy vzdáleny i více než 100 metrů, také při ručním zadávání souřadnic trasy se nepředpokládá, že bude uživatel v rovných úsecích trasy vytvářet souřadnice každých 30 metrů. Proto by byl výpočet navigačních bodů u těchto tras, pomocí algoritmu použitého v mobilní aplikaci, značně nepřesný.

Před výpočtem navigačních bodů se u všech souřadnic trasy vypočítá směr k následující souřadnici. Samotný výpočet navigačních bodů probíhá následujícím způsobem (Výpis 5.11). Nejprve se do proměnné `next` uloží index první souřadnice vzdálené od aktuální souřadnice více než 30 metrů. V rovných úsecích je to většinou následující souřadnice, v zatáčkách to může být index i více než 10. souřadnice. Poté se, pomocí metody `compareBearing(double, double)`, vypočítá rozdíl směrů mezi aktuální souřadnicí a souřadnicí na indexu, který je uložen v proměnné `next`, tento rozdíl směrů se uloží do proměnné `bearing`. Pokud je hodnota v proměnné `bearing` větší než 25, nachází se v daném úseku zatáčka doprava, pro ověření, zda je tomu opravdu tak, se ještě porovná směr aktuální souřadnice se směrem souřadnice uložené na indexu `next + 1`. Pokud je tento rozdíl směrů větší než 25, u všech souřadnic v daném úseku trasy se nastaví navigační značka na `RIGHT`. Jako aktuální souřadnice je nastavena poslední souřadnice tohoto úseku. Pokud je hodnota v proměné `bearing` menší než -25, nachází se v daném úseku zatáčka doleva, také zde se ověří zda je také rozdíl směrů, aktuální souřadnice a souřadnice uložené na indexu `next + 1`, menší než -25. Pokud ano, nastaví se u všech souřadnic v daném úseku trasy navigační značka na `LEFT`. Je-li hodnota v proměnné `bearing` v rozmezí mezi -25 a 25, je daný úsek trasy rovný. V tomto případě se u všech

souřadnic v daném úseku trasy nastaví navigační značka na `STRAIGHT`.

```
int index = 0; int next; bool end;
while (true) {
  next = 1; end = false;
  next = //index první souřadnice vzdálené od aktuální souřadnice více než 30 metrů
  float bearing = compareBearing(coords[index].bearing, coords[index + next].bearing);
  if (bearing > 25) {
    bearing = compareBearing(coords[index].bearing, coords[index + next + 1].bearing);
    if (bearing > 25) {
      for (int i = 0; i < next; i++)
        coords[index + i].navigation = Sounds.RIGHT;
      index += next;
    } else {
      for (int i = 0; i < next; i++)
        coords[index + i].navigation = Sounds.STRAIGHT;
      index += next;
  }} else if (bearing < -25) {
    bearing = compareBearing(coords[index].bearing, coords[index + next + 1].bearing);
    if (bearing < -25) {
      for (int i = 0; i < next; i++)
        coords[index + i].navigation = Sounds.LEFT;
      index += next;
    } else {
      for (int i = 0; i < next; i++)
        coords[index + i].navigation = Sounds.STRAIGHT;
      index += next;
  }} else {
    for (int i = 0; i < next; i++)
      coords[index + i].navigation = Sounds.STRAIGHT;
    index += next;
}}
```

Výpis 5.11: Výpočet navigačních bodů

Pomocí tohoto algoritmu je přiřazena navigační značka každé souřadnici trasy. Proto je nutné ještě jednou souřadnice trasy projít a vymazat navigační značky u souřadnic, na kterých se nenachází změna směru. Navigační značka se nechá pouze u první souřadnice trasy a také u souřadnic, které mají navigační značku jinou než předchozí souřadnice. Příklad: Prvních 5 souřadnic trasy bude mít navigační značky: rovně, rovně, vlevo, vlevo, vpravo. Po promazání budou navigační značky vypadat takhle: rovně, nic, vlevo, nic, vpravo.

5.9 Určení krajů, ve kterých se trasa nachází

U každé trasy, vytvořené pomocí informačního systému nebo nahrané z mobilní aplikace, se při ukládání do databáze informačního systému automaticky vypočítá, ve kterých krajích se daná trasa nachází. Souřadnice krajů jsou uloženy v databázi informačního systému v tabulce `region_coordinate`. Netestují se všechny souřadnice trasy, bylo by to značně neefektivní. Testují se pouze první souřadnice trasy a poté souřadnice ležící ve vzdálenosti 15-ti kilometrů od té předchozí. Ověření, zda se souřadnice trasy nachází v některém z krajů, probíhá pomocí paprskového algoritmu.

Paprskový algoritmus[19] určuje, zda se souřadnice nachází v určitém kraji na základě počtu průsečíků (Výpis 5.12). Ze souřadnice je vedena polopřímka, která určuje počet průsečíků s hranami kraje. Pokud je počet průsečíků lichý, souřadnice se nachází v určitém kraji. Je-li počet sudý (i roven nule), souřadnice se v daném kraji nenachází. Průsečík je započítán pouze v případě, pokud je jedna ze souřadnic hrany vlevo od polopřímky a druhá ze souřadnic na nebo vpravo od polopřímky. Kolineární hrany nejsou započítávány. Hledány jsou pouze průsečíky ležící nad testovanou souřadnicí.

```
int cn = 0;
for (int i = 0; i < coords.Count − 1; i++) {
  if (((coords[i].longitude <= c.longitude) && (coords[i + 1].longitude > c.longitude))
   || ((coords[i].longitude > c.longitude) && (coords[i + 1].longitude <= c.longitude))) {
    float vt =(c.longitude − coords[i].longitude)/(coords[i + 1].longitude − coords[i].longitude);
    if (c.latitude < coords[i].latitude + vt * (coords[i + 1].latitude − coords[i].latitude))
      cn++;
}}
if ((cn & 1) == 0)
  return false;
else
  return true;
```

Výpis 5.12: Implementace paprskového algoritmu

Do proměnné `cn` se ukládá počet průsečíků. V cyklu jsou procházeny všechny hrany kraje, u každé hrany se testuje, zda je jedna souřadnice hrany vlevo od polopřímky a druhá souřadnice na nebo vpravo od polopřímky. Pokud ano, do proměnné `vt` se uloží přesná hodnota zeměpisné délky (longitude), ve které je hrana polopřímkou protnuta. Poté se ověří, zda je souřadnice, kterou testujeme, pod hranou. Pokud ano, zvýší se počet průsečíků o hodnotu 1. Pokud je po projití všech hran kraje počet průsečíků lichý, je vrácena hodnota true. Pokud je počet průsečíků sudý, je vrácena hodnota false.

6 Komunikace mobilní aplikace s informačním systémem

V této části bude popsána veškerá komunikace mobilní aplikace s informačním systémem. Budou zde podrobně popsány všechny operace, které probíhají mezi mobilní aplikací a informačním systémem, jsou to operace přihlášení na server a synchronizace tras, načtení tras ze serveru, nahrání trasy na server, načtení souřadnic trasy ze serveru, nahrání souřadnic trasy na server a nahrání informací o jízdě na server.

6.1 Přihlášení na server a synchronizace tras

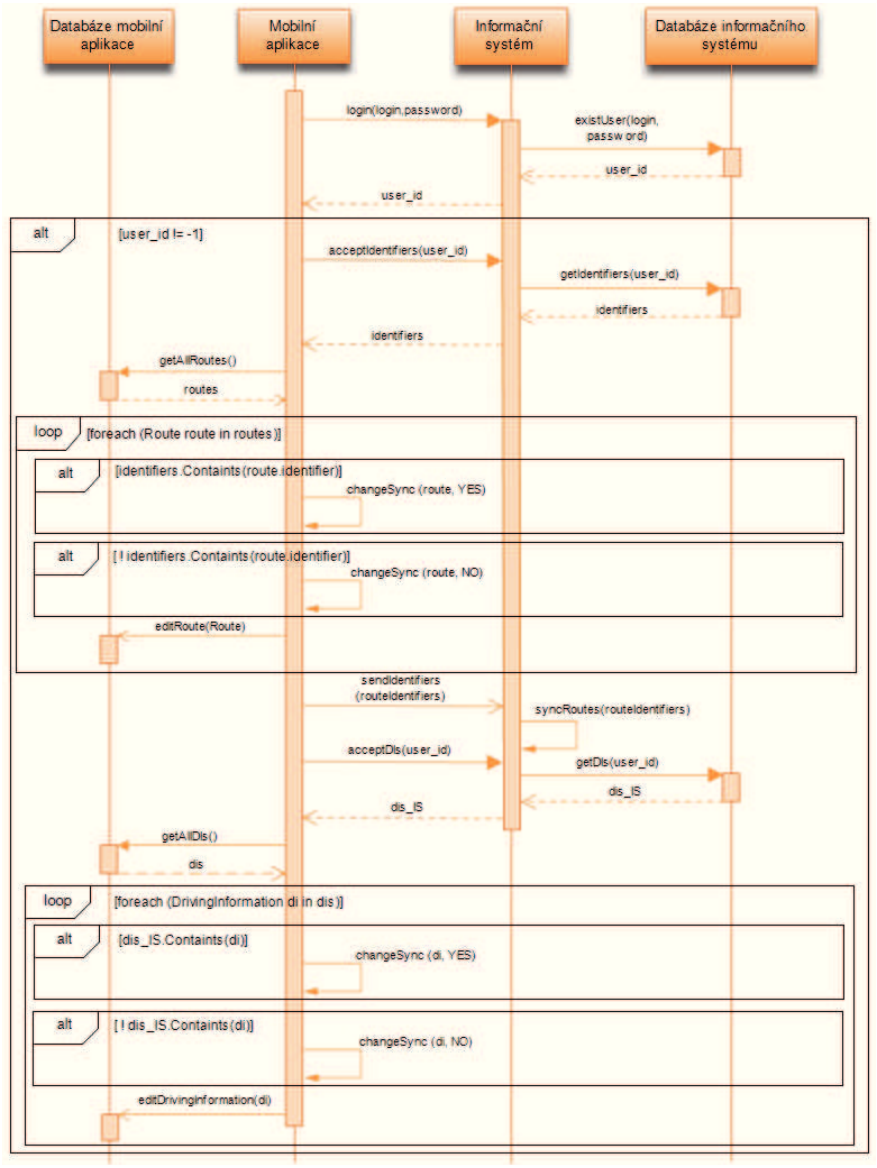
Tato operace slouží pro přihlášení na uživatelský účet, založený na informačním systému, pomocí mobilní aplikace. Pokud uživatel zadá správné přihlašovací údaje, proběhne také synchronizace tras a informací o jízdě, které jsou nahrány v mobilní aplikaci, s trasami a informacemi o jízdě, které má uživatel uložen ke svému účtu na informačním systému. Synchronizací je myšlena pouze změna informací o synchronizaci u jednotlivých tras a informací o jízdě, nejsou načítány žádné trasy ze serveru, ani nahrávány žádné trasy na server, k těmto úkonům slouží jiné operace, které budou popsány níže.

Posloupnost jednotlivých operací, které probíhají při přihlašování na server a následné synchronizaci tras a informací o jízdě, je znázorněna pomocí sekvenčního diagramu (Obrázek 6.1).

Nejprve se z mobilní aplikace pomocí WCF služby `login(string, string)` pošle přihlašovací jméno a heslo na informační systém. Zde se ověří, zda je v databázi uložen uživatel s těmito přihlašovacími údaji. Pokud je takový uživatel nalezen, metoda `login(string, string)` vráti ID uživatele. Pokud není v databázi uložen žádný uživatel s těmito přihlašovacími údaji, metoda `login(string, string)` vrátí hodnotu -1. Pokud je v databázi informačního systému nalezen uživatel se zadanými přihlašovacími údaji, je zavolána WCF služba `acceptIdentifiers(int)`, která vrací pole, které obsahuje identifikační značky tras, které jsou uloženy v databázi informačního systému k účtu tohoto uživatele, dále jsou také načteny všechny trasy uloženy v databázi mobilní aplikace. Poté se u každé trasy zjišťuje, zda je její identifikační značka obsažena v poli identifikačních značek, které vrátila metoda `acceptIdentifiers(int)`. Pokud ano, změní se informace o synchronizaci této trasy na YES, a pokud ne, změní se informace o synchronizaci na NO. Nakonec je tato změna uložena do databáze.

Poté jsou na informační systém pomocí WCF služby `sendIdentifiers(string[])` poslány identifikační značky všech tras uložených v databázi mobilní aplikace. A poté proběhne změna informací o synchronizaci u tras, uložených v databázi informačního sy-

tému k účtu přihlášeného uživatele, tato synchronizace probíhá stejným způsobem jako synchronizace na mobilním zařízení.



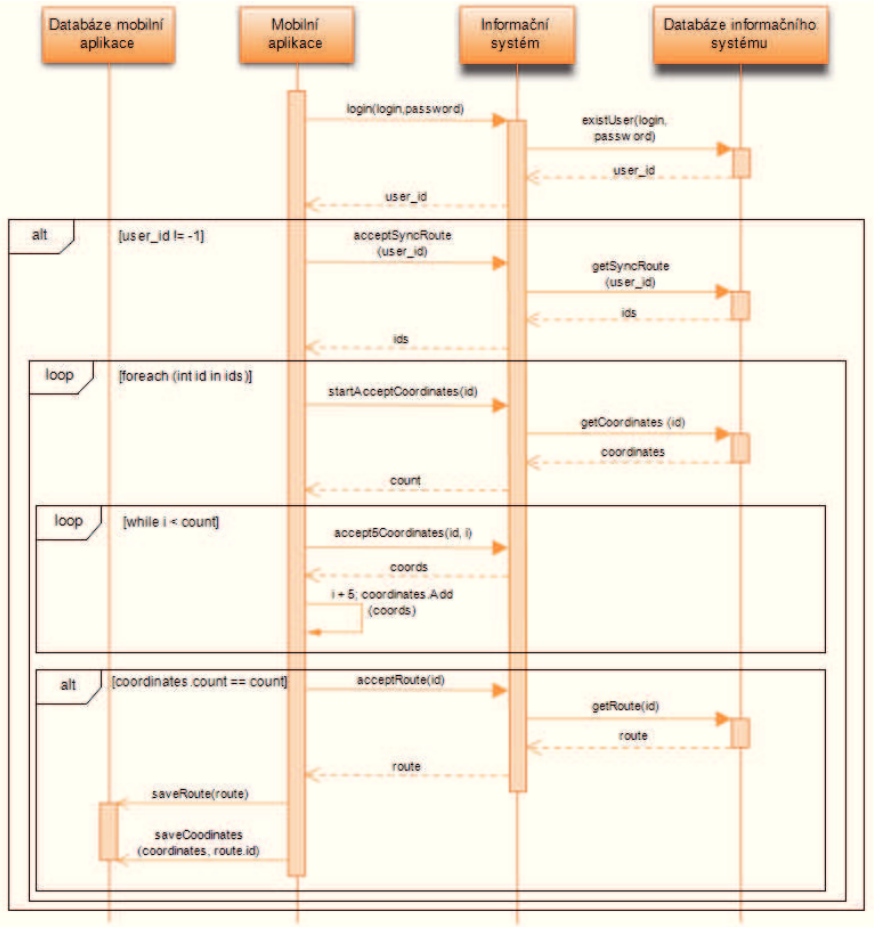
Obrázek 6.1: Sekvenční diagram - Přihlášení na server a synchronizace tras

Nakonec probíhá synchronizace informací o jízdě, tato synchronizace probíhá stejně jako synchronizace tras, s tím rozdílem, že informace o jízdě nemají identifikační značku. Proto se u synchronizace informací o jízdě porovnávají identifikační značky tras, ke kterým jsou informace o jízdě naměřeny, datum, kdy byly tyto informace o jízdě naměřeny, a čas jízdy. Je velice nepravděpodobné, že budou ve stejný den naměřeny dvoje informace

o jízdě ke stejné trase a se stejným časem jízdy. Synchronizace informací o jízdě na informačním systému není nutná, protože informace o jízdě lze pouze nahrávat z mobilní aplikace na informační systém, nikoliv naopak.

6.2 Načtení tras ze serveru

Tato operace slouží pro načtení tras, které jsou na informačním systému označeny jako čekající na nahrání na mobilní zařízení. Pro úspěšné provedení této operace je nutné být přihlášen ke svému účtu pomocí mobilní aplikace. Posloupnost jednotlivých operací je znázorněna pomocí sekvenčního diagramu (Obrázek 6.2).



Obrázek 6.2: Sekvenční diagram - Načtení tras ze serveru

Nejprve proběhne ověření přihlašovacích údajů, které se provádí pomocí WCF služby `login (string, string)`. Poté je zavolána WCF služba `acceptSyncRoute(int)`, která vrací pole obsahující ID tras, které jsou uloženy v databázi informačního systému k

účtu přihlášeného uživatele, označené jako čekající na nahrání na mobilní zařízení. Poté je spuštěn cyklus, který postupně prochází všechna ID tras, které byly vráceny metodou acceptSyncRoute(int). Pro každé ID trasy jsou nejprve z informačního systému načteny souřadnice a až poté samotná trasa.

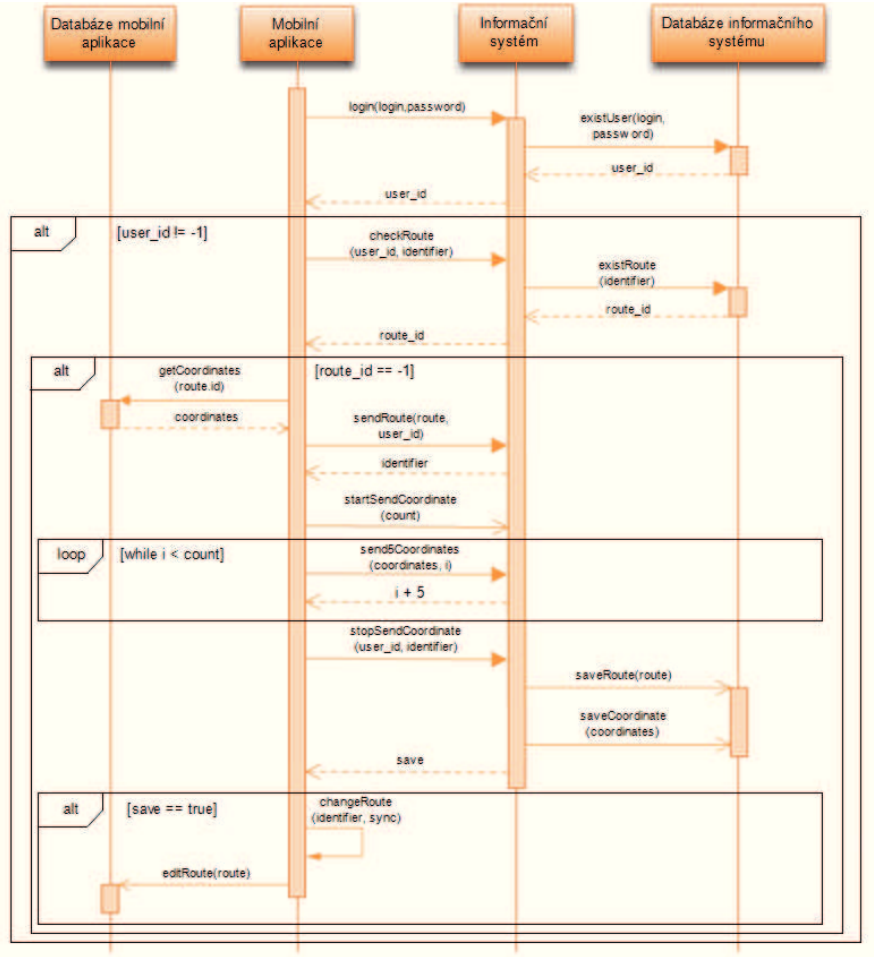
Načítání souřadnic probíhá ve dvou krocích. V prvním kroku je zavolána WCF služba `startAcceptCoordinates(int)`, po zavolání této WCF služby se z databáze informačního systému načtou příslušné souřadnice, služba poté vrací počet těchto souřadnic. V druhém kroku je opakovaně volána WCF služba `accept5Coordinates(int, int)`, která načte 5 souřadnic z informačního systému, tato WCF služba je volána dokud je druhý parametr menší než počet souřadnic vrácený v prvním kroku. Druhý parametr je index první posílané souřadnice, který je před každým opakovaným voláním zvýšen o hodnotu 5. Poté se ověří, zda byly načteny všechny souřadnice trasy. Pokud ano, je pomocí WCF služby `acceptRoute(int)` načtena z informačního systému samotná trasa. Nakonec je trasa, a poté i načtené souřadnice, uložena do databáze mobilní aplikace.

6.3 Nahrání trasy na server

Tato operace slouží pro nahrání trasy, uložené v mobilní aplikaci, na informační systém. Pro úspěšné provedení této operace je nutné být přihlášen ke svému účtu pomocí mobilní aplikace. Posloupnost jednotlivých operací je znázorněna pomocí sekvenčního diagramu (Obrázek 6.3)

Nejdříve proběhne ověření přihlašovacích údajů, které se provádí pomocí WCF služby `login (string, string)`. Poté se pomocí WCF služby `checkRoute(int, string)` ověří, zda trasu, která má být poslána na informační systém, již uživatel nemá uloženou ve svých trasách. Prvním parametrem WCF služby je ID uživatele a druhým parametrem je identifikační značka trasy, služba vrací ID trasy, pokud je trasa s touto identifikační značkou uložena k účtu přihlášeného uživatele, a nebo hodnotu -1, pokud taková trasa není v databázi nalezena.

Pokud není posílaná trasa uložena v databázi informačního systému, načtou se z databáze mobilní aplikace souřadnice trasy, a proběhne nahrání trasy na informační systém. Nejdříve je pomocí WCF služby `sendRoute(int, Route)` na informační systém poslána samotná trasa, tato služba vrací identifikační značku trasy. Trasy, které jsou naměřeny mobilní aplikací a nejsou nahrány na informační systém, nemají identifikační značku, tato značka je každé trase přiřazena až při nahrání trasy na server, pokud nahrávaná trasa již identifikační značku má, značka se nezmění.



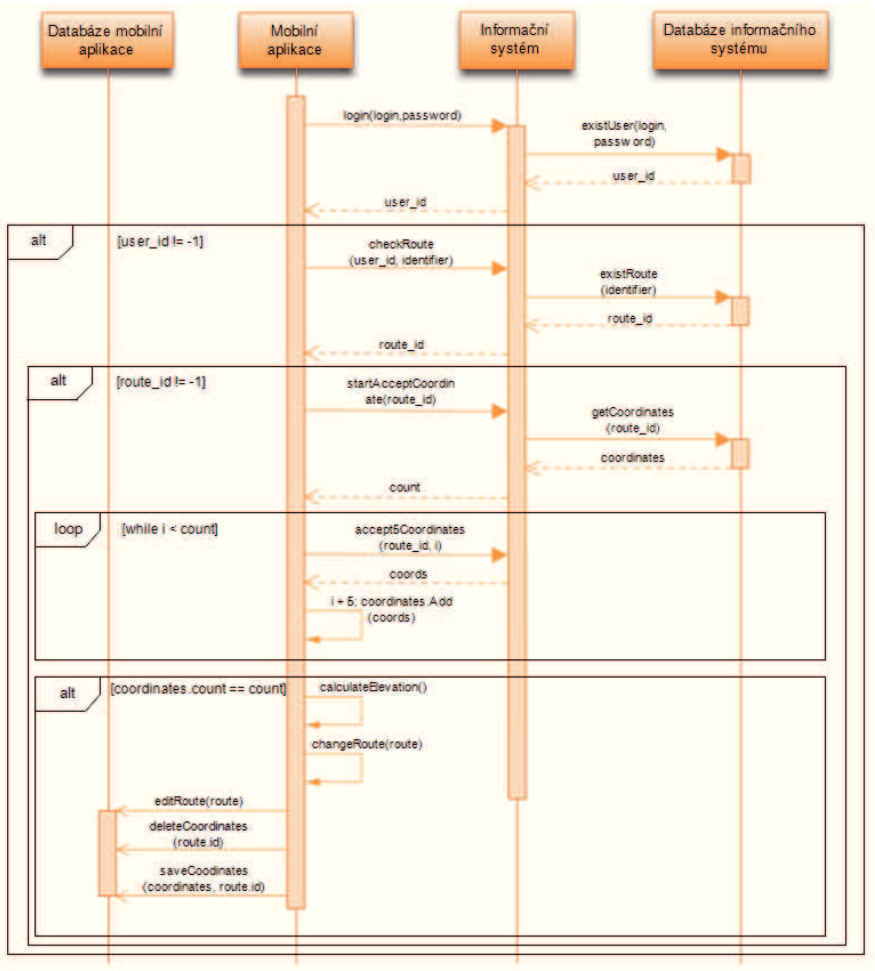
Obrázek 6.3: Sekvenční diagram - Nahrání trasy na server

Poté proběhne nahrání souřadnic trasy, toto nahrávání probíhá ve třech krocích. V prvním kroku je pomocí WCF služby `startSendCoordinate(int)` na informační systém poslán počet souřadnic. V druhém kroku je opakovaně volána WCF služba `send5Coordinates(ArrayList<RouteCoordinate>, int)`, která pošle 5 souřadnic z mobilní aplikace na informační systém. Tato WCF služba je volána dokud je druhý parametr menší než počet posílaných souřadnic. Druhý parametr je index první posílané souřadnice trasy, který je po každém opakovaném volání této WCF služby zvýšen o hodnotu 5. V posledním kroku je zavolána WCF služba `stopSendCoordinate(int, string)`. Po zavolání této služby se na informačním systému ověří, zda byly poslány všechny souřadnice. Pokud ano, je trasa spolu se souřadnicemi uložena do databáze. Při úspěšném uložení trasy na informačním systému, se upraví trasa na mobilním zařízení, ke trase se přidá identifikační značka, pokud trasa tuto značku ještě nemá přiřazenou, a

také se změní informace o synchronizaci.

6.4 Načtení souřadnic trasy ze serveru

Tato operace slouží pro načtení souřadnic trasy z informačního systému do mobilní aplikace. Pro úspěšné provedení této operace je nutné být přihlášen ke svému účtu pomocí mobilní aplikace. Posloupnost jednotlivých operací je znázorněna pomocí sekvenčního diagramu (Obrázek 6.4).



Obrázek 6.4: Sekvenční diagram - Načtení souřadnic trasy ze serveru

Nejprve proběhne ověření přihlašovacích údajů, které se provádí pomocí WCF služby `login (string, string)`. Poté se pomocí WCF služby `checkRoute(int, string)` provede kontrola, zda je trasa, ke které jsou uloženy načítané souřadnice, uložena v databázi informačního systému k účtu přihlášeného uživatele. Pokud jsou splněny obě tyto

podmínky proběhne načtení souřadnic.

Načítání souřadnic probíhá ve dvou krocích. V prvním kroku je zavolána WCF služba `startAcceptCoordinates(int)`, po zavolání této WCF služby se z databáze informačního systému načtou příslušné souřadnice, služba poté vrací počet těchto souřadnic. V druhém kroku je opakovaně volána WCF služba `accept5Coordinates(int, int)`, která načte 5 souřadnic trasy, tato WCF služba je volána dokud je druhý parametr menší než počet souřadnic vrácený v prvním kroku. Druhý parametr je index první posílané souřadnice, který je před každým opakovaným voláním zvýšen o hodnotu 5. Poté se ověří, zda byly načteny všechny souřadnice trasy. Pokud ano, je z načtených souřadnic vypočítána nová délka trasy, nastoupané metry, naklesané metry a průměrný sklon trasy. Poté se tyto hodnoty uloží do databáze mobilní aplikace, dále se z databáze vymažou staré souřadnice trasy a jsou místo nich uloženy nové souřadnice trasy.

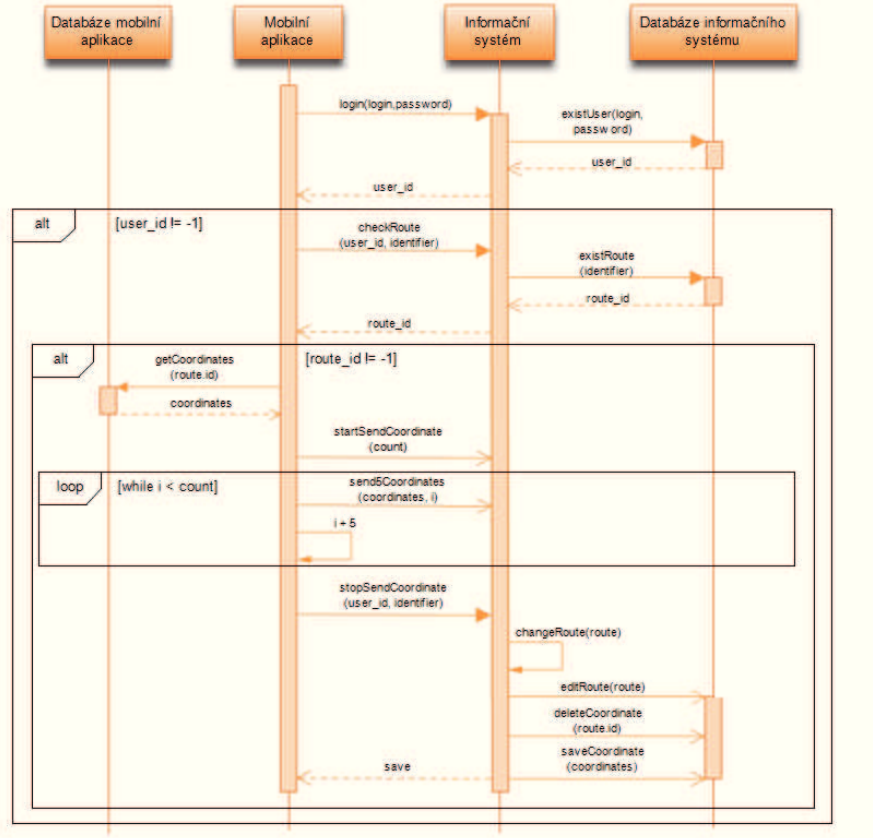
6.5 Nahrání souřadnic trasy na server

Tato operace slouží pro nahrání souřadnic trasy z mobilní aplikace na informační systém. Pro úspěšné provedení této operace je nutné být přihlášen ke svému účtu pomocí mobilní aplikace. Posloupnost jednotlivých operací je znázorněna pomocí sekvenčního diagramu (Obrázek 6.5).

Nejprve proběhne ověření přihlašovacích údajů, které se provádí pomocí WCF služby `login (string, string)`. Poté se pomocí WCF služby `checkRoute(int, string)` provede kontrola, zda je trasa, ke které mají být nahrány souřadnice, uložena v databázi informačního systému k účtu přihlášeného uživatele. Pokud jsou splněny obě tyto podmínky, proběhne nahrání souřadnic na informační systém.

Nahrávání souřadnic probíhá ve třech krocích. V prvním kroku je pomocí WCF služby `startSendCoordinate(int)` na informační systém poslán počet souřadnic. V druhém kroku je opakovaně volána WCF služba `send5Coordinates(ArrayList <RouteCoordinate>, int)`, která pošle 5 souřadnic z mobilní aplikace na informační systém. Tato WCF služba je volána dokud je druhý parametr menší než počet posílaných souřadnic. Druhý parametr je index první posílané souřadnice trasy, který je po každém opakovaném volání této WCF služby zvýšen o hodnotu 5. V posledním kroku je zavolána WCF služba `stopSendCoordinate(int, string)`. Po zavolání této služby se na informačním systému ověří, zda byly poslány všechny souřadnice. A pokud ano, je z poslaných souřadnic vypočítána nová délka trasy, nastoupané metry, naklesané metry a průměrný sklon trasy. Poté se tyto hodnoty uloží do databáze informačního systému,

dále se z databáze vymažou staré souřadnice trasy a jsou místo nich uloženy nové souřadnice trasy.

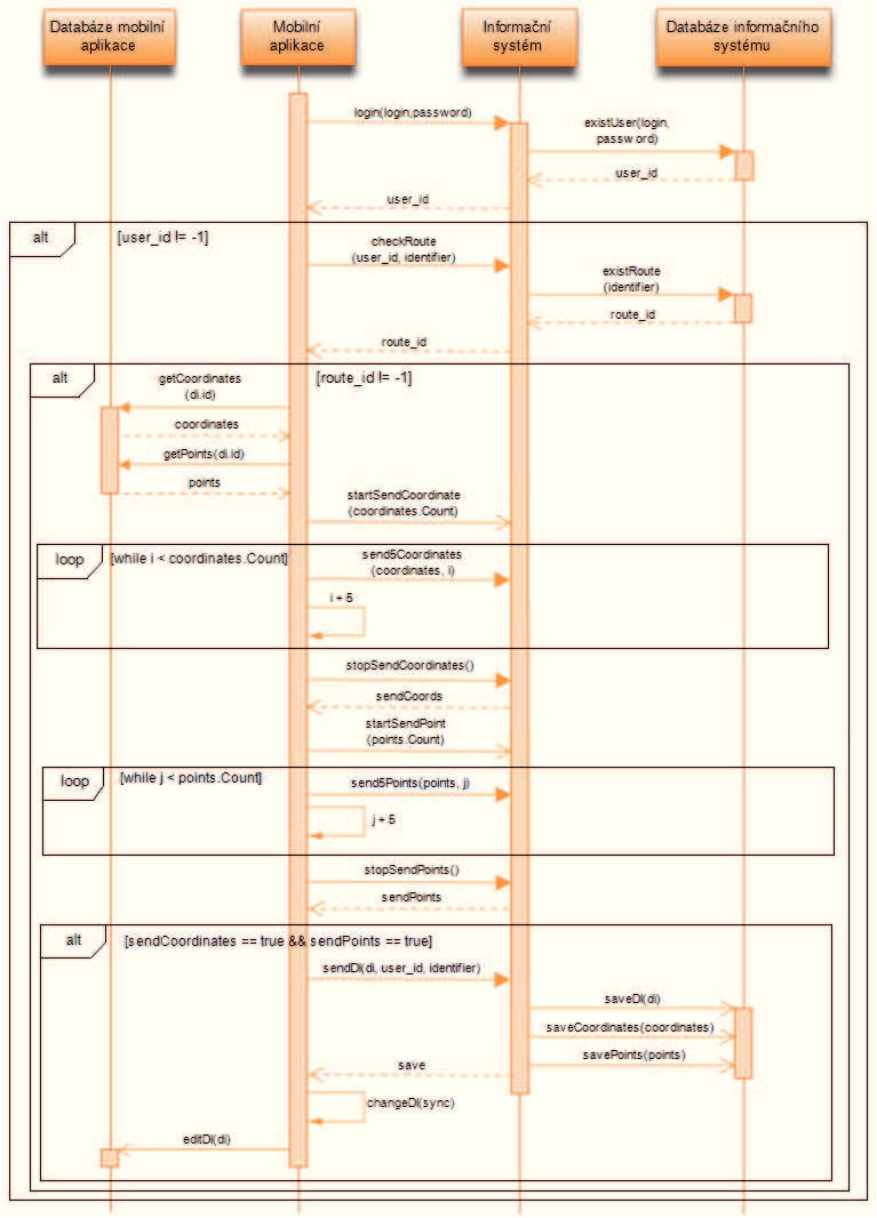


Obrázek 6.5: Sekvenční diagram - Nahrání souřadnic trasy na server

6.6 Nahrání informací o jízdě na server

Tato operace slouží pro nahrání informací o jízdě z mobilní aplikace na informační systém. Pro úspěšné provedení této operace je nutné být přihlášen ke svému účtu pomocí mobilní aplikace. Posloupnost jednotlivých operací je znázorněna pomocí sekvenčního diagramu (Obrázek 6.6).

Nejprve proběhne ověření přihlašovacích údajů, které se provádí pomocí WCF služby `login (string, string)`. Poté se pomocí WCF služby `checkRoute(int, string)` provede kontrola, zda je trasa, ke které mají být nahrány informace o jízdě, uložena v databázi informačního systému k účtu přihlášeného uživatele. Pokud jsou splněny obě tyto podmínky proběhne nahrání informací o jízdě na informační systém.



Obrázek 6.6: Sekvenční diagram - Nahrání informací o jízdě na server

Nejprve jsou z databáze mobilní aplikace načteny souřadnice, uložené k informacím o jízdě, a také body grafu rychlosti v závislosti na čase jízdy. Na informační systém jsou první poslány souřadnice, uložené k informacím o jízdě, poté body grafu rychlosti v závislosti na čase jízdy, a nakonec samotné informace o jízdě. Souřadnice i body grafu rychlosti jsou na informační systém posílány stejným způsobem, jakým se posílají souřadnice trasy při nahrávání trasy na server. Proto ho zde již nebudu znovu podrobně popisovat.

Pokud se na informační systém úspěšně nahrají všechny souřadnice i body grafu rychlosti, jsou na informační systém nahrány samotné informace o jízdě. Poté jsou informace o jízdě, souřadnice i body grafu rychlosti uloženy do databáze informačního systému. V mobilní aplikace se u informací o jízdě změní informace o synchronizaci na YES.

7 Závěr

Cílem této diplomové práce bylo naimplementovat funkční cyklonavigační systém skládající se ze dvou částí, z mobilní aplikace určenou pro platformu Android a informačního systému.

Při implementaci mobilní aplikace i informačního systému bylo důležité navrhnout databázi tak, aby se ukládala pouze důležitá data a práce s databází byla pokud možno co nejrychlejší. Mobilní aplikace i informační systém jsou strukturovány tak, aby poskytovaly co největší přehled v uložených datech. Pro využití všech funkcí systému se uživatel musí v informačním systému zaregistrovat. Poté má uživatel možnost vytvářet vlastní trasy, nahrávat trasy z mobilní aplikace, nebo si do svých tras přidávat trasy veřejné. U všech tras, které má uživatel uloženy ve svých trasách, může upravovat souřadnice i navigační body, a také lze tyto trasy nahrát do mobilní aplikace.

V mobilní aplikaci lze, stejně jako na informačním systému, u jednotlivých tras a informací o jízdě zobrazit naměřené hodnoty do několika přehledných grafů. Samozřejmostí je vykreslení trasy na mapě s vyznačením nadmořské výšky, aktuální rychlosti nebo sklonu trasy. Na mapě lze také zobrazit navigační body trasy. Mobilní aplikaci lze také využít jako plnohodnotnou navigaci. Mobilní aplikace zvládá navigaci po trase pomocí hlasových hlášení i pomocí grafického zobrazení. Navíc je mobilní aplikace schopna v režimu jízdy přehrávat skladby, které si uživatel přidá do playlistu.

Mobilní aplikace byla testována na mobilních telefonech Samsung Galaxy Young a T-Mobile G1, také známý pod označením HTC Dream. Na obou telefonech běží aplikace plynule. Aplikaci lze stáhnout přes informační systém, aplikace není dostupná na Android Marketu. Informační systém běží na internetové adrese *cyklonavigace.aspone.cz*. Všechny funkce informačního systému byly testovány v internetových prohlížečích Mozilla Firefox, Google Chrome a Internet Explorer. V internetových prohlížečích Google Chrome a Internet Explorer běží všechny funkce bez problémů. V internetovém prohlížeči Mozilla Firefox nefunguje úprava souřadnic a navigačních bodů trasy. Důvodem je použití knihovny Artem.Google, pomocí které probíhá práce s mapou. V internetovém prohlížeči Mozilla Firefox jsou špatně zpracovávány události zpracovávané na serveru. Bohužel jsem na tuto skutečnost přišel až v době testování informačního systému, a neměl jsem dostatek času na odstranění tohoto nedostatku.

8 Literatura

[1] STEELE James; TO Nelson, *The Android Developer's Cookbook: Building Applications with the Android SDK*. 1. edition. UK: Addison-Wesley Professional, 2010. 400 s. ISBN 978-0321741233

[2] GARGENTA Marko, *Learning Android.* 1. edition. UK: O'Reilly Media, 2011. 268 s. ISBN 978-1449390501

[3] BURNETTE Ed, *Hello, Android: Introducing Google's Mobile Development Platform*. 3. edition. UK: Pragmatic Bookshelf 2010. 300s. ISBN 978-1934356562

[4] MURPHY Mark, *Beginning Android*. 1. edition. UK: Apress, 2009. 384 s. ISBN 978-1430224198

[5] *Android Developers* [online], c2013 [cit. 05-01-2013]. Dustupné z WWW: <http://developer.android.com>

[6] MEIER Reto, *Professional Android 4 Application Development (Wrox Programmer to Programmer).* 3. edition. UK: Wrox, 2012. 864 s. ISBN 978-1118102275

[7] FRIESEN Jeff, *Learn Java for Android Development*. 1. edition. UK: Apress, 2010. 656 s. ISBN 978-1430231561

[8] MEDNIEKS Zigurd; DORNIN Laird; MEIKE G. Blake; NAKAMURA Masumi, *Programming Android: Java Programming for the New Generation of Mobile Devices*. 2. edition. UK: O'Reilly Media, 2012. 566 s. ISBN 978-1449316648

[9] WEI Jason, *Android Database Programming*. 1. edition. UK: Packt Publishing, 2012. 212 s. ISBN 978-1849518123

[10] BEAULIEU Alan, *Learning SQL*. 2. edition. UK: O'Reilly Media, 2009. 338 s. ISBN 978-0596520830

[11] ESPOSITO Dino, *Programming Microsoft ASP.NET 4*. 1. edition. UK: Microsoft Press, 2011. 992 s. ISBN 978-0735643383

[12] *Microsoft Developer Network* [online], c2013 [cit. 05-01-2013]. Dustupné z WWW: <http://msdn.microsoft.com>

[13] MACDONALD Matthew; FREEMAN Adam, *Pro ASP.NET 4 in C# 2010*. 4. edition. UK: Apress, 2010. 1616 s. ISBN 978-1430225294

[14] SVENNERBERG Gabriel, *Beginning Google Maps API 3 (Expert's Voice in Web Development)*. 2. edition. UK: Apress, 2010. 328 s. ISBN 978-1430228028

[15] *Codeplex, GoogleMap Control* [online], c2013 [cit. 07-04-2013]. Dustupné z WWW: <http://googlemap.codeplex.com/>

[16] *The Google Elevation API* [online], c2013 [cit. 07-04-2013]. Dustupné z WWW: <https://developers.google.com/maps/documentation/elevation/>

[17] *Google Maps Image API* [online], c2013 [cit. 07-04-2013]. Dustupné z WWW: <https://developers.google.com/maps/documentation/staticmaps/>

[18] *Google Chart Tools* [online], c2013 [cit. 07-03-2013]. Dustupné z WWW: <https://developers.google.com/chart/>

[19] *Geometrical algorithms* [online], c2013 [cit. 07-03-2013]. Dustupné z WWW: <http://geomalgorithms.com/a03-_inclusion.html>